ANALYSE COMPARATIVE DE L'UTILISATION DE L'APPRENTISSAGE PROFOND SUR DES IMAGES SATELLITAIRES

par

Charles Authier

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 19 novembre 2020

Le 19 novembre 2020

le jury a accepté le mémoire de Monsieur Charles Authier dans sa version finale.

Membres du jury

Professeur Pierre-Marc Jodoin
Directeur de recherche
Département d'informatique

Professeur Christian Desrosiers
Évaluateur externe
Département de Génie logiciel et des TI de l'ÉTS

Professeur Maxime Descoteaux
Président-rapporteur
Département d'informatique

Sommaire

L'analyse d'images satellites est un domaine de la géomatique permettant de nombreuses observations par rapport à la terre. Une étape importante de toute observation est d'identifier le contenu de l'image. Cette étape est normalement effectuée à la main, ce qui coûte temps et argent. Avec l'avènement des réseaux de neurones profonds, des GPUs à forte capacité de calculs et du nombre croissant de données satellitaires annotées, les algorithmes apprenants sont désormais les outils les plus prometteurs pour l'analyse automatique d'images satellitaires.

Ce mémoire présente une étude préliminaire de l'application des réseaux à convolution sur des images satellites, ainsi que deux nouvelles méthodes devant permettre d'entraîner des réseaux de neurones a l'aide de données satellitaires pauvrement annotées. Pour cela, on a utilisé deux bases de données de l'*international society for photogrammetry and remote sensing* comprenant 40 images étiquetées de six classes. Les deux atouts majeurs de ces bases de données sont la grande variété des canaux composant leurs images, ainsi que les lieux différents (et donc contextes) où ces images ont été acquises. Par la suite, nous présenterons des résultats empiriques à plusieurs questions d'ordre pratique en lien avec les performances attendues des réseaux de neurones profonds appliqués à l'imagerie satellitaire. Vers la fin du rapport, nous présenterons deux techniques permettant de combiner plusieurs ensembles de données, et ce, grâce à des étiquettes de classes hiérarchiques.

Mots-clés: apprentissage machine ; apprentissage profond ; classification ; segmentation ; télédétection ; imagerie satellite ; imagerie aérienne.

Remerciements

Premièrement, merci à Pierre-Marc Jodoin, mon directeur de recherche, pour son appui et ses idées durant ces deux années.

Merci à Christian Desrosiers pour son appui et ses idées.

Merci à tous les membres du VITAL avec qui j'ai pu échanger opinions et bons moments au cours de mon séjour au laboratoire (Clément, Frédéric, Philippe, Carl, Faezeh, Antoine, Jon, Sarah, Audrey, Nathan, Youssef et Thierry).

Merci à la compagnie *Urthecast* de m'avoir aidé à l'obtention de la bourse ENGAGE.

Un gros merci à toute l'équipe de Ressources Naturelles Canada pour le soutien financier et les données.

Finalement, je remercie ma famille et mes amis pour leur soutien depuis le début, même s'ils ne comprennent pas tout ce que je fais.

Abréviations

NASA National Aeronautics and Space Administration (en français l'Administration nationale de l'aéronautique et de l'espace).

ASE Agence Spatiale Européenne (en anglais : **ESA** - European Space Agency).

FAQ Foire aux questions.

CCD Dispositif à transfert de charges (en anglais : *Charge-Coupled Device*).

CMOS Complementary Metal-Oxide-Semiconductor.

RSO Radar à Synthèse d'Ouverture (en anglais : **SAR** - Synthetic Aperture Radar).

LiDAR Light detection and ranging ou Laser detection and ranging, dépendement du domaine de recherche (pas d'équivalent français).

RVB Rouge-Vert-Bleu (en anglais : **RGB** - Red-Green-Blue).

IR Infra-Rouge.

IDVN Indice de Différence de Végétation Normalisé (en anglais : **NDVI** - Normalized Difference Vegetation Index).

IDEN Indice de Différence d'Eau Normalisé (en anglais : **NDWI** - Normalized Difference Water Index).

RAG Réseaux Adverses Génératifs (en anglais : **GAN** - Generative Adversarial Network).

ISPRS International Society for Photogrammetry and Remote Sensing [1].

1. http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html

Table des matières

Liste des figures

Liste des tableaux

Liste des tableaux

Introduction

Depuis près de 30 ans, de nombreuses agences spatiales, telles que la NASA et l'ESA, ont mis à disposition une partie des images satellites prises lors de leurs missions. De plus, 1992 marque l'arrivée du *"Land Remote Sensing Policy Act"*[2], permettant aux compagnies privées de lancer leurs propres satellites d'imagerie à des fins commerciales. Au fil des années, la superficie terrestre couverte par ces satellites a augmenté et les capteurs et les méthodes d'acquisitions ont évolué, donnant accès à des milliards d'images de la planète.

Aujourd'hui, l'usage de ces images est très diversifié, allant du domaine militaire à la gestion du risque, en passant par l'agriculture et l'aménagement du territoire. Les informations transmises par ces images sont très utiles, mais l'extraction de ces informations est complexe et demande beaucoup de ressources. C'est ici que les réseaux de neurones entrent en jeux.

Imagerie satellitaire et problématiques

L'imagerie satellitaire, aussi appelée télédétection, est la science qui vise l'acquisition d'informations sur la surface terrestre, via un satellite. Pour ce faire, un satellite capte les ondes provenant de la surface de la Terre qu'il observe, et les enregistre sous forme de signal dans le temps. L'ordinateur à l'intérieur du capteur procède ensuite à un traitement et à une analyse de ce signal, changeant selon l'usage donné au satellite. Aujourd'hui, le recours aux capteurs satellitaires en orbite, ainsi qu'à l'analyse qu'ils

2. H.R.6133 - Land Remote Sensing Policy Act of 1992 : https://www.congress.gov/bill/102nd-congress/house-bill/6133

fournissent, donne accès à une foule d'informations, transformant la façon dont les humains voient leur planète.

Il y a quelques années, il était complexe d'exécuter des tâches de classification ou d'identification avec ce type de données. En effet, certaines problématiques, telles que la haute dimensionnalité des données et la faible disponibilité d'images étiquetées, rendaient difficile la résolution de ce type de problème. Récemment, avec l'arrivée de l'apprentissage profond, l'utilisation des données satellites est devenue beaucoup plus simple, et de nouvelles solutions à ces problèmes ont pu voir le jour. Ce type d'apprentissage offre donc de nouvelles opportunités pour la recherche et le développement d'outils, permettant un traitement plus rapide et plus précis des images satellites.

Solutions proposées et contributions

Ce que l'on propose est d'utiliser le concept de transfert de connaissances en apprentissage profond pour mieux généraliser à de nouveaux jeux de données similaires, mais avec des différences quand même importantes. On cherche à créer une nouvelle architecture qui facilitera le transfert de connaissance avec une annotation faible sur un nouveau jeu de données.

Nous étudierons entre autres de nouvelles architectures hiérarchiques dédiées au transfert de connaissances et propres à l'imagerie satellitaire. À travers cette étude, on va chercher à mesurer l'influence de l'annotation faible sur le transfert de connaissances et aussi sa stabilité à travers les entraînements. De plus, on va chercher à voir le lien entre les classes et leurs proportions lors du transfert de connaissances.

Le chapitre 1 présentera l'univers de l'imagerie satellitaire, présentant les capteurs et leurs utilisations actuelles. Il sera suivi par le chapitre 2 qui explique les fondements de l'apprentissage automatique. Ensuite, au chapitre 3, on présente des résultats et des méthodes complémentaires au projet principal. Pour finir, le chapitre 4 expose l'utilisation d'images satellitaires partiellement annotées dans des réseaux d'apprentissage profond.

Chapitre 1

Imagerie satellitaire

La superficie de la Terre est d'environ 510 072 000 km^2, dont 70.8% (361 132 000 km^2) sont des étendues d'eau, contre seulement 29.2% (148 940 000 km^2) recouverts de terre[69]. Cette large différence entre les proportions apporte son lot de problématiques. En effet, toutes les zones couvertes sont observées avec une relativement grande résolution. De plus, traiter toute l'information, utile ou non, requiert un temps considérable.

1.1 Modalités, capteurs et collecte de données satellite

Les capteurs utilisés dans les satellites sont similaires à ceux retrouvés dans les appareils photo et cellulaires, à quelques différences près. Pour bien comprendre ces différences, il faut d'abord comprendre la physique derrière ceux-ci.

Tout d'abord, les capteurs les plus communs sont les "dispositif à transfert de charges" (CCD) et les "Complementary Metal-Oxide-Semiconductor" (CMOS). Ceux-ci sont des composants électroniques photosensibles, servant à convertir un rayonnement électromagnétique, tel que montré dans le tableau 1.1, en un signal analogique. Après une série de procédés, le résultat est une image numérique.

Nom de la région spectrale	Longueur d'onde
Rayon Gamma	<0.03 nm
Rayon X	0.03 à 30 nm
Ultraviolet	30 à 380 nm
Visible	380 à 780 nm
Proche infrarouge	0.78 à 1.4 μm
Infrarouge à moyenne longueur d'onde	1.4 à 3 μm
Infrarouge lointain	3 à 1000 μm
Micro-ondes et radar	0.001 à 100 cm
Radio	>1 m

tableau 1.1: Régions importantes du spectre électromagnétique [1] [2]

Saviez-vous que ?
Les poissons rouges peuvent voir dans l'infrarouge et que les bourdons peuvent voir dans l'ultraviolet.

Pour comprendre le fonctionnement d'un capteur, il suffit de voir ce dernier comme un puits. Pendant un certain temps, le puits est exposé à une quantité de rayonnement électromagnétique, appelée photons. Certains de ces photons auront l'énergie suffisante pour exciter des électrons du capteur, qui seront accumulés dans le puits. À la fin du temps d'exposition, le puits est vidé, créant un courant électrique ensuite transformé en signal électrique. Plus il y aura de photons sur le détecteur, plus il y aura d'électrons dans le puits, et plus le signal sera intense.

La plupart des capteurs sont malheureusement souvent limités aux spectres visible et proche infrarouge. Cette limitation est principalement due à l'effet photo-électrique associée au matériau dont le capteur est fait. En effet, les longueurs d'onde des photons pouvant exciter des électrons sont déterminées, entre autres, par le matériau qui

1. https://www.rncan.gc.ca/sciences-terre/geomatique/imagerie-satellitaire-photos-aeriennes/imagerie-satellitaire-produits/ressources-educatives/14624
2. https://fr.wikipedia.org/wiki/Spectre_%C3%A9lectromagn%C3%A9tique

compose le capteur. Les matériaux les plus courants, étant sensibles aux spectres visibles et proches infrarouges uniquement, l'imagerie qu'ils permettent est également limitée à ces spectres. Il existe évidemment certains capteurs permettant d'acquérir l'information des autres régions spectrales, mais pour ce faire, il faut utiliser des matériaux photosensibles réactifs aux longueurs d'onde ciblées.

Pour finalement créer une image, les appareils photo possèdent plusieurs matrices 2D de puits, captant chacune des couleurs différentes (attention, ici le tri des couleurs n'est pas fait par l'utilisation de différents matériaux, mais par différentes tensions électriques appliquées aux bornes des puits). Pour ce qui est d'un satellite, l'idée d'avoir un appareil de mesure composé de cinq ou six capteurs n'est pas rentable. La solution est d'utiliser une seule matrice 2D, mais composée de plusieurs capteurs 1D, captant chacun une différente longueur d'onde. Cette matrice effectuera un balayage en continu pour former une image 2D. Afin d'optimiser l'efficacité de chaque capteur, des prismes de verres sont utilisés pour séparer les photons de la bonne longueur d'onde et de les diriger vers le bon capteur 1D, comme l'illustre l'album de Pink Floyd, "Dark Side of the Moon", avec la lumière visible.

1.1.1 Radar à synthèse d'ouverture

Un **SAR** est un radar imageur qui effectue un traitement des données reçues afin d'améliorer la résolution en azimut. On parle de synthèse d'ouverture, contrairement à un radar à visée latérale classique, d'où le nom de ce type de système. Le capteur du radar est relativement petit et il émet un signal au sol qui est la résultante d'une amplitude et d'une phase. Tous les échos générés par tous les points couverts par l'impulsion émise sont le fruit de l'intégral (au sens mathématique du terme) de l'espace couvert. Le signal reçu est donc un point de la transformée de Fourier de la zone couverte. Comme le radar se déplace, il reçoit d'autres points(échos) de cette transformée[80]. En enregistrant ces points et en effectuant la transformée inverse, il est possible de reconstituer le relief en deux dimensions du sol (2D) et en appliquant les procédés d'interférométrie on obtient la troisième dimension (3D)[43].

Ce type de capteur permet de cartographier la surface terrestre, malgré les nuages, de jour comme de nuit. Il s'agit donc d'un capteur extrêmement utile pour la sur-

veillance des changements qui s'opèrent dans la masse continentale et les zones côtières. Au Canada, la constellation de satellites **RADARSAT**[3][4] est utilisée pour l'analyse des eaux de surface et milieux humides, des glaces des lacs et des rivières d'eau douce, de la déformation de la surface et stabilité des pentes, ainsi que l'évaluation multithématique.

1.1.2 LiDAR

La télédétection par laser est une technique de mesure de distance basée sur l'analyse des propriétés d'un faisceau de lumière renvoyé vers son émetteur. Le principe physique du LiDAR requiert généralement l'utilisation d'un laser impulsionnel. La distance est donnée par la mesure du délai entre l'émission d'une impulsion et la détection d'une impulsion réfléchie, connaissant la vitesse de la lumière. Ce type de mesure est très utile pour déterminer les changements d'élévations au sol, les données recueillies sont appelées nuage de points. Ces résultats peuvent être ensuite utilisés pour déterminer le développement urbain dans les villes, l'effacement des sols, la hauteur des arbres, etc. Cependant, très peu de satellites sont équipés de ce type de capteur. L'alternative est d'utiliser un avion, mais les coûts associés à ces acquisitions sont très élevés pour couvrir une zone avec une résolution correspondant à la résolution d'une image couleur. Bien que très utile, ce type de données est rare.

1.1.3 Visible et infrarouge

Les capteurs RGB fonctionnent dans le domaine du visible et de l'infrarouge. Ils mesurent la quantité de rayonnement réfléchi (ou émis) depuis la terre et de l'atmosphère sus-jacente. Les capteurs RGB sont les plus nombreux et les plus utilisés puisqu'ils sont les plus facilement interprétables au moment de l'acquisition. Cependant, comme mentionné plus haut, ce type de capteur mesure la quantité de rayonnement réfléchi ou émis par un objet. Une combinaison des différents canaux permet la visualisation et l'interprétation de différentes modalités dans l'image.

3. RADARSAT-1 (inactif) :http://www.asc-csa.gc.ca/fra/satellites/radarsat1/default.asp
4. RADARSAT-2 (actif) :http://www.asc-csa.gc.ca/fra/satellites/radarsat2/default.asp

Par exemple, la combinaison appelée "fausses couleurs" est représentée par la combinaison des canaux autre que "R-G-B", "R-G-B → G-B-R" et autres, mais dans le cas de la télédétection, la combinaison "IR-R-G" est plus communément utilisée. La raison est que la végétation qui normalement apparaît en vert, dans le cas de fausses couleurs elle apparaîtra en ton de rouge, car la végétation reflète davantage de rayonnement dans l'infrarouge (c.f. fig. 1.1).

(a) **Vaihingen** - IRRG (b) **Potsdam** - RGB

figure 1.1: Exemple d'images provenant du jeux de donnée **ISPRS**.

1.1.4 Hyperspectrale et multispectrale

Une image RGB est constituée de trois bandes spectrales. Comme son nom l'indique, le multispectrale et l'hyperspectrale sont simplement des images composées de plus de trois bandes. L'imagerie multispectrale réfère généralement à une image constituée de quatre à dix bandes spectrales, alors que l'imagerie hyperspectrale tourne autour de cent à mille bandes. Ces bandes sont situées dans les spectres du visible et de l'infrarouge.

Le principal avantage de l'imagerie hyperspectrale est qu'un spectre entier est acquis à chaque point, l'opérateur n'a donc pas besoin de connaître préalablement l'échantillon. De plus, le post-traitement permet d'exploiter toutes les informations disponibles à partir du jeu de données. L'imagerie hyperspectrale peut également tirer parti des relations spatiales entre les différents spectres d'un pixel voisin, permettant ainsi des modèles spectrospatiaux plus élaborés pour une segmentation et une classification plus précises de l'image[68][71]. Il est de même pour l'imagerie multispectrale, mais avec un spectre plus restreint. Cela permet toutefois de mettre de l'avant des relations souvent utilisées en agriculture. De plus, l'acquisition de données multispectrales permet la manipulation des bandes pour obtenir des indices de végétation ou de point d'eau, cette partie ne sera toutefois pas traitée dans ce mémoire.

Les principaux inconvénients sont le coût et la complexité d'acquisition des données. Des ordinateurs rapides et performants, des détecteurs sensibles et de grandes capacités de stockage de données sont nécessaires pour analyser des données multispectrales et hyperspectrales. Une capacité de stockage est nécessaire dû aux cubes hyperspectraux qui sont de grands ensembles de données multidimensionnelles. Tous ces facteurs augmentent considérablement le coût d'acquisition et de traitement des données hyperspectrales. En outre, l'un des obstacles auxquels se heurtent les chercheurs est de traiter les images sur le satellite et de ne transmettre vers la terre que les images les plus importantes, car la transmission et le stockage de cette quantité de données se révèlent difficiles et coûteux. Selon de nombreux chercheurs, le potentiel de l'imagerie hyperspectrale n'a pas encore été pleinement exploité.

1.2 Bases de données

Cette section sera consacrée aux bases de données dédiées à la *Classification*, la *Détection* et la *Segmentation*. La *classification* consiste à assigner une étiquette de la classe à une image entière. La *détection* correspond à la localisation, à l'aide d'un encadré, un ou plusieurs objets dans une image pour ensuite les *classifier* en leur assignant une étiquette de classe. La *Segmentation* quant à elle, a pour objectif d'assigner une étiquette de classe à chaque pixel de l'image. Dans le cas où il y a plus

qu'une étiquette de classe, il s'agit alors d'une segmentation sémantique.

Cette section fait un sommaire des bases de données les plus utilisées pour ces trois tâches. À noter que ces bases de données sont propres à leurs créateurs et donc très différentes les unes des autres.

1.2.1 Classification

Ce problème est également appelé *classification d'images*, les bases de données d'images satellitaires dédiées à la classification ont pour particularité de contenir des "*crops*" d'images à haute résolution. Ainsi, chaque petite image ne contient qu'un seul objet : nuage, voiture, maison, stationnement, etc. Le tableau 1.2 présente les jeux de données les plus populaires en classification d'images satellitaires. Parmi lesquels on retrouve de nombreuses classes bien différentes, mais ceux que l'on voit revenir le plus souvent sont ceux qui classifient l'environnement qui compose l'image (ex. ville, désert, zone agricole, etc.).

Nom	Modalité	Nombre de Classes	Nombre d'Images	Dimension	Résolution(m)/pixel
UC Merced Land Use Dataset[91]	RGB	21	2 100	256×256	0.3048
BigEarthNet[81]	13 bandes	43	590 326	120×120 60×60 20×20	10m bandes 20m bandes 60m bandes
WiDS Datathon 2019 [5]	RGB	2	20 000	256×256	3
So2Sat LCZ42[99]	8 bandes Sentinel-1 10 bandes Sentinel-2	17	400 000	32×32	10
CONACYT (UAV)[57]	RGB	2	17 000	32×32	inconnue
C-CORE Iceberg Classifier Challenge [6]	SAR (2 bandes)	2	5 625	75×75	inconnue
Functional Map of the World Challenge[17]	4 bandes QuickBird-2/GeoEye-1 8 bandes WorldView-2/WorldView-3	63	132 716 boîtes uniques	224×224	0.3
EuroSAT[32]	3 et 16 bandes Sentinel-2	10	27 000	64×64	10
Planet : Understanding the Amazon from Space [7]	4 bandes (RGB-NIR)	17	150 000	64×64	5
RESISC45[16]	RGB	45	31 500	256×256	∼30 à 0.2
Deepsat[8]	4 bandes (RGB-NIR)	6	400 000	28×28	1

tableau 1.2: Bases de données pour la *Classification* d'images satellites.

5. https://www.kaggle.com/c/widsdatathon2019/overview

La section 1.3.3 présentera différents algorithmes de classification, ainsi que leurs applications dans la vie quotidienne.

1.2.2 Détection d'objets

La détection d'objets est une tâche de localisation et classification d'objets dans une image. Il est possible qu'une image haute résolution puisse contenir plusieurs objets nécessitant d'être localisés et classifiés. Il s'agit d'une tâche plus ardue que la simple classification d'images, car il existe souvent plusieurs objets dans une même image. Souvent, les techniques développées pour la classification des images avec localisation sont utilisées et démontrées pour la détection d'objets (c.f. tableau 1.3). Contrairement à la classification, la détection d'objets a généralement des images de plus grande résolution, car elles doivent pouvoir clairement illustrer plusieurs objets dans son entièreté. La classe qui revient le plus souvent et qui est la plus demandée sur les forums, est la classe "Voiture", l'application la plus évidente est militaire, mais cela peut avoir beaucoup d'autres avantages pour le suivi des véhicules d'urgence et l'optimisation de leur trajet.

Nom	Modalité	Nombre de Classes	Nombre de boîte de détection	Résolution(m)/pixel
DOTA[89]	RGB	15	188 000	multiple
DIUx xView 2018[47]	3 et 8 bandes	60	1 million	0.3
NIST DSE Plant Identification with NEON Remote Sensing Data [8]	RGB hyperspectrale nuage de point	3	plusieurs millions	0.25 1.0 inconnue
The Rio De Janeiro Points of Interest Dataset[11]	3 et 8 bandes	460	120 155	0.5
Cars Overhead With Context (Aérienne)[63]	RGB	1	32 000	0.15
Airbus Ship Detection Challenge [9]	3 et 8 bandes	1	131 000	0.3/0.5

tableau 1.3: Bases de données pour la *Détection d'objets* sur des images satellite.

6. https://www.kaggle.com/c/statoil-iceberg-classifier-challenge/overview
7. https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/overview

1.2.3 Segmentation sémantique

Contrairement à la détection d'objets qui implique l'utilisation d'un cadre de sélection pour identifier des objets, la segmentation d'objets classifie chaque pixel d'une image. De façon générale, la *segmentation d'images* fait référence à l'assignation de tous les pixels d'une image en différentes catégories d'objets (c.f. tableau 1.4).

Nom	Modalité	Nombre de Classes	Nombre d'Images	Dimension	Résolution(m)/pixel
xView 2 Building Damage Asessment Challenge[26]	3 et 8 bandes	2	inconnue	variable	0.3
Microsoft Building Footprints Canada & USA [10]	RGB	2	3 million	256×256	0.3
Spacenet Buildings Round 1-5 [11]	3 et 8 bandes	2	inconnue	variable	0.3/0.5
Spacenet Roads[11]	3 et 8 bandes	2	inconnue	variable	0.3/0.5
38-Cloud : A Cloud Segmentation Dataset[61]	4 bandes	2	17 601	384×384	0.3/0.5
DLRSD[77]	RGB	17	2 100	256×256	inconnue
DSTL [12]	3, 8 et 16 bandes	10	450	variable	variable
Indian Pines [13]	200 bandes	16	1	145×145	inconnue
Salinas scene[13]	224 bandes	16	1	512×217	3.7
Pavia Centre[13] Pavia University[13]	103 bandes 102 bandes	9	1	1096×1096 610×610	inconnue
Kennedy Space Center[13]	176 bandes	13	1	512×614	18
Vaihingen [14] Potsdam[14]	5 bandes 4 bandes	5	16 24	variables 6000×6000	0.09 0.05

tableau 1.4: Bases de données pour la *Segmentation sémantique* d'images satellite.

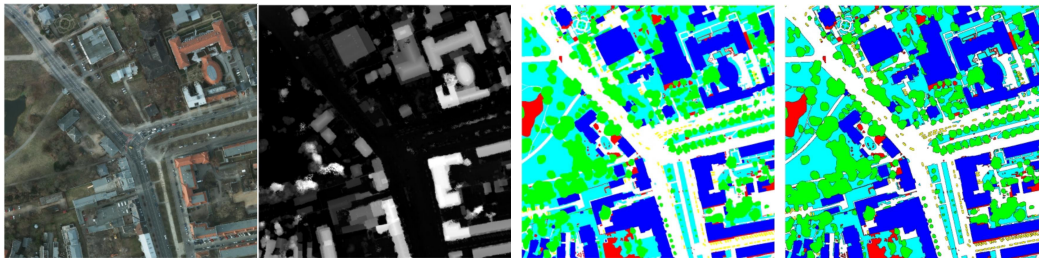
8. https://www.ecodse.org/
9. https://www.kaggle.com/c/airbus-ship-detection/overview
10. https://github.com/microsoft/USBuildingFootprints(USA) https://github.com/Microsoft/CanadianBuildingFootprints(CANADA)
11. https://spacenetchallenge.github.io/datasets/datasetHomePage.html
12. https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection/overview
13. http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes
14. http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html

ISPRS (*International Society for Photogrammetry and Remote Sensing*)

Pour cette maîtrise, la base de données utilisée est **ISPRS**[14] (*International Society for Photogrammetry and Remote Sensing*). Celle-ci est gratuitement disponible et comporte des images de plusieurs régions et modalités. Puisque ce projet porte sur la segmentation, le sous-ensemble de la base de données sélectionnée est "*2D Semantic Labeling*"[14]. Ce dernier est composé de deux villes Allemandes : *Vaihingen* et *Potsdam*. Les images **ISPRS** ont été annotées avec précision et utilisées à de nombreuses occasions par la communauté scientifique ce qui en fait une base de données la mieux reconnue en télédétection. La communauté dernière **ISPRS** met à disposition un rapport détaillé des résultats dits "*state-of-the-art*" résultant de leurs données, sur leur site internet.

Les bases de données des deux villes comportent des images orthorectifiées avec les canaux infrarouge, rouge et vert. La ville de **Vaihingen** est constituée de 33 grandes images de taille variable extrait d'une plus grande image de la ville avec une résolution au sol(en anglais, *ground sample distance*, GSD) de 9 cm. De ces 33 images, 16 ont été manuellement annotées. Un exemple de Vaihingen est représenté à la figure 1.1a.

Pour ce qui est de la ville de **Potsdam**, le canal bleu est aussi compris. Elle comporte 38 images d'une résolution de 6000×6000pixels^2 avec une résolution au sol de 5 cm. 24 d'entre elles ont été manuellement annotées. Un exemple de Potsdam est représenté aux figures 1.1b et 1.2.



(a) RGB (b) GSD (c) Avec Contours (d) Sans Contours

figure 1.2: Exemple de modalités de la ville de **Potsdam** ((a) et (b)), ainsi que les deux vérités terrain, avec (c) et sans (d) contours autour des objets.

La figure 1.2 montre les modalités et les annotations fournies par la base de données ISPRS. La vérité terrain des deux villes comporte six classes : surface imperméables (blanc), bâtiments (bleu), basse végétation (cyan), arbres (vert), voitures (jaune) et obstructions/background (rouge). La catégorie "obstructions/background" inclut les surfaces avec de l'eau (présent dans seulement deux images) et des objets variables qui semblent différents des autres catégories (e.g., poubelles, terrains de tennis, piscines, etc.).

1.3 Tâches usuelles et difficultés inhérentes

1.3.1 Mise en registre

La mise en registre d'images est une méthode permettant d'aligner au moins deux images capturées par différents capteurs, à différents moments ou selon différents points de vue[100][93]. Il s'agit d'une étape fondamentale pour de nombreuses tâches d'analyse par télédétection. Selon Zitova et Flusser[100], la mise en registre d'images comprend les quatre étapes suivantes : extraction de caractéristiques, correspondance des caractéristiques, estimation du modèle de transformation et ré-échantillonnage de l'image. Étant donné que l'apprentissage profond est un système entièrement basé sur les données, il peut donc apprendre automatiquement les caractéristiques des images ce qui en fait un outil idéal pour aligner des d'images de télédétection.

La plupart des méthodes de mise en registre d'images utilisant l'apprentissage profond reposent sur un réseau siamois[10][60][29][37][87]. L'idée de base de ces méthodes est d'utiliser un réseau de neurones profonds composé de deux parties :

- Extraction des fonctionnalités de correctifs d'image en formant un réseau siamois ou pseudo-siamois.
- Mesurer la similarité entre ces caractéristiques pour la correspondance d'images.

Les méthodes adversaires peuvent aussi être appliquées à la mise en registre d'images [59][36]. Ces méthodes se traduisent d'abord par la transformation d'une image pour ressembler le plus à l'autre image, ce qui permet aux deux images d'avoir une in-

tensité ou des informations similaires. Ensuite, le réseau procède à l'extraction des caractéristiques et à la correspondance entre les deux images générées artificiellement, ce qui améliore efficacement les performances de la correspondance d'images.

Outre la mise en registre d'une image sur une autre, certaines méthodes sont également utilisées pour la mise en registre d'images sur des cartes lorsque ceux-ci ne sont pas fournis avec leurs géo-informations. L'article de Zampieri et al.[94] présente une chaîne de réseaux de neurones spécifiques à l'échelle, pour l'enregistrement des cartes de bâtiments ainsi que des polylignes routières sur des images aériennes. Par la suite, l'article de Girard et al.[25] présente certains réseaux d'apprentissage multitâche qui ont pour caractéristique d'améliorer les performances d'alignement.

Cependant, à l'heure actuelle, il n'existe pas de base de données dédiée à la mise en registre des images de télédétection et les échantillons d'apprentissage doivent être réalisés manuellement. En raison de la diversité des données de télédétection (résolution différente, acquisition à des moments différents, acquisition avec des modalités différentes), il est difficile et laborieux d'établir de vastes ensembles de données pour la mise en registre d'images.

1.3.2 L'affichage panchromatique et super résolution

L'affichage panchromatique (*pansharpening* en anglais) consiste à améliorer la résolution spatiale des données multispectrales en les fusionnant avec des données caractérisées par des informations spatiales plus précises. C'est un cas particulier du problème plus général de la super-résolution utilisée pour des images RGB.

Plusieurs satellites d'imageries enregistrent plusieurs bandes spectrales avec différentes résolutions spatiales. Par exemple, des capteurs multispectraux à multirésolutions sont présents dans les satellites MODIS, VIIRS, ASTER, Worldview-3 et Sentinel-2. Les avantages de ces capteurs sont que les différentes bandes spectrales sont enregistrées quasi simultanément avec un éclairage et des conditions atmosphériques identiques et avec un angle d'acquisition identique (après une légère correction). Les résolutions entre les bandes spectrales de tout instrument sont uniques et diffèrent

généralement par un facteur d'environ 2 à 6. Par exemple, les satellites Sentinel-2 ont leurs bandes R-G-B et une granularité GSD de 10m, leurs images proches infrarouges ont une granularité GSD de 20m et leurs images infrarouges à moyenne longueur d'onde ont une granularité GSD de 60m. Il est donc impossible de les superposer avec la même résolution et de les traiter de la même façon[48].

Les raisons pour lesquelles on enregistre les images à différentes résolutions sont généralement la restriction de stockage, la restriction de bande passante pour la transmission satellite-terre et des bandes conçues à des fins spécifiques ne nécessitant pas une résolution spatiale élevée (les corrections atmosphériques par exemple). Dans certains cas, il y a aussi l'amélioration du rapport signal sur bruit (en anglais SNR) via des pixels plus grands.

Pourtant, il est souvent souhaitable d'avoir toutes les bandes disponibles à la plus haute résolution spatiale. En général, les images basse résolution sont redimensionnées avec des méthodes simples et rapides, mais naïves comme des interpolations bilinéaire ou bicubique. Cependant, ces méthodes retournent des images floues avec peu d'informations supplémentaires. Des méthodes plus sophistiquées, comme l'apprentissage profond, tentent de faire mieux en tirant avantage des détails spatiaux et spectraux, grâce aux bandes haute résolution disponible[48]. L'objectif de la super résolution et le *pansharpening* est de dépasser les méthodes traditionnelles de reconstruction, tout en préservant les informations spectrales des bandes originales.

Le plus gros avantage de ces méthodes, c'est l'abondance des données, puisque celles-ci ne nécessitent pas de vérité terrain. Il suffit de manuellement réduire la résolution de l'image tout en gardant le même nombre de pixels et entraîner un réseau à reproduire l'image d'origine via un certain facteur. Le seul inconvénient est qu'une fois que les bandes spectrales sont toutes à la même résolution spatiale, il n'existe aucune méthode pour valider la précision spectrale de ces dernières.

1.3.3 Classification de scènes et détection d'objets

Il est important de faire la différence entre la classification et la détection d'objets, car ce sont des applications similaires en matière de télédétection.

La classification des scènes est définie comme une procédure permettant de déterminer les catégories d'images à partir de nombreuses images[101](exemple : des scènes agricoles, des scènes de forêt et des scènes de plage). Les données d'apprentissage sont une série d'images étiquetées avec une classe. La détection d'objets, quant à elle, vise à détecter et localiser différents objets dans une même scène(exemple : des avions[98], des voitures[23] et des villages urbains[52]). La grande différence entre ces deux cas est que dans un cas on a des images de faible résolution ne contenant qu'un seul objet et dans l'autre on a des images de haute résolution contenant un très grand nombre d'objets Les données d'apprentissage sont les pixels d'une fenêtre de taille fixe ou variable, car dans un cas on a des images de faible résolution ne contenant qu'un seul objet et dans l'autre on a des images de haute résolution contenant un très grand nombre d'objets.

Toutefois, les applications pratiques en télédétection rencontrent davantage de types d'objets et de données, ayant des résolutions différentes avec des tailles différentes. Par conséquent, la difficulté est de concevoir des algorithmes d'apprentissage efficaces pour surmonter les difficultés rencontrées par des objets variés à différentes échelles[21].

1.3.4 Segmentation

La segmentation sémantique d'images est une tâche qui consiste à classer chaque pixel d'une image à partir d'un ensemble prédéfini de classes. Dans l'exemple de la figure 1.3, différentes entités sont classées selon un numéro propre à une classe.

L'objectif est de prendre une image de taille $L \times H \times C$ (où C correspond au nombre de canaux de l'image d'entrée) et de générer une matrice $L \times H$ contenant les étiquettes correspondantes à la classe prédite pour chaque pixel. La segmentation sémantique est différente de la détection d'objets, car elle ne prédit pas de boîte englobante autour des objets. Toutefois, dans ce cas-ci, il n'est pas question de distinction entre différentes

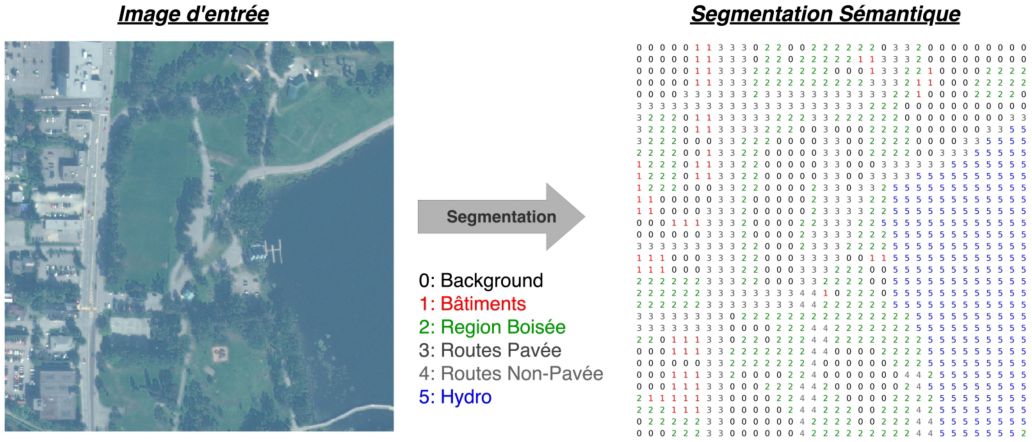


figure 1.3: Illustration du principe de la segmentation sémantique

instances du même objet, même si cela est possible. Par exemple, il pourrait y avoir plusieurs voitures dans la scène et toutes auraient la même étiquette.

Afin d'effectuer une segmentation sémantique, une compréhension de niveau supérieur de l'image est requise. L'algorithme doit comprendre les objets présents ainsi que les pixels qui correspondent à l'objet. La segmentation sémantique est l'une des tâches essentielles pour une compréhension complète de la scène présente dans l'image.

L'intérêt pour la segmentation sémantique des images satellites s'est accru ces dernières années. Tout d'abord, les images sont segmentées en régions homogènes (des segments également appelés objets) représentant un groupe relativement homogène de pixels en sélectionnant les critères d'échelle, de forme et de compacité souhaitées. Dans un deuxième temps, un processus de classification est appliqué à ces objets. Étant donné que ce procédé offre la possibilité d'exploiter la fonctionnalité de système d'information géographique (SIG), tel que l'incorporation du contexte spatial ou de la forme de l'objet dans la classification, elle fournit un cadre permettant de surmonter les limitations des méthodes de classification d'images classiques à base de pixels. Les applications à la cartographie peuvent être utilisées pour identifier des glissements de terrain[50], la couverture terrestre et l'utilisation des terres[51], ainsi

qu'à la détection de changements[18].

La qualité de la délimitation de la segmentation des objets a une influence directe sur la précision de la classification des images d'entrées. Au cours des dernières décennies, de nombreuses techniques de segmentation d'images ont été développées et appliquées à l'analyse d'images de télédétection. Cependant, comme Hay et al.[28] ont souligné, le véritable défi consiste à définir les paramètres de segmentation appropriés pour les méthodes traditionnelles (généralement basées sur l'homogénéité spectrale, la taille ou les deux) pour des objets/images de tailles, de formes et de modalité variés, composant une scène, afin de générer des segments.

Les algorithmes de segmentation traditionnelles multi-résolution[76] est probablement l'alternatif le plus répandu aux réseaux de neurones à des fins de la délimitation d'objets. Le paramètre de résolution est utilisé pour contrôler l'hétérogénéité interne (spectrale) des objets résultants et est donc corrélé à leur taille moyenne. En d'autres termes, une valeur plus grande de l'échelle permet une hétérogénéité interne plus importante, ce qui augmente le nombre de pixels par objet et vice versa.

Depuis lors, l'apprentissage profond a montré son pouvoir de représentation pour les images satellitaires. D'une part, par rapport aux méthodes non basées sur les réseaux de neurones qui impliquent une ingéniosité humaine considérable pour la conception des fonctionnalités, les réseaux de neurones profonds extraient directement les caractéristiques des données. En comparaison aux techniques traditionnelles d'apprentissage de représentation d'entités (par exemple, "*Sparse Coding Based Feature RepresentationMethod for Remote Sensing Images*"[65]), l'apprentissage profond peut en extraire une représentation beaucoup plus précise. En outre, les caractéristiques des couches supérieures du réseau de neurones affichent les propriétés d'abstraction sémantique.

Malgré les progrès réalisés jusqu'à présent, certains problèmes sont spécifiques à la segmentation sémantique des images satellites, notamment : la difficulté à modéliser la grande variété d'objets géographiques dans les images à très haute résolution spatiale [85], résoudre le problème du débalancement des classes causé par de petits objets

présents en faible quantité [56] et tirer avantage des bandes multi-spectrales ou des données multimodales[78][4][58].

1.3.5 Détection de changement

La détection de changement consiste à utiliser deux images satellitaires, prises à des moments différents pour détecter les changements causés par des phénomènes naturels (sécheresse, inondations, etc.) ou par l'homme (l'ajout d'une nouvelle route, démolition de vieux bâtiments, etc.). La détection de changement est sûrement le domaine de recherche le moins exploité en apprentissage profond[7], principalement dû au manque de données annotées.

En règle générale, le processus de détection de changement peut être divisé en trois grandes parties :

- Pré-traitement (rectification géométrique, corrections radiométriques et atmosphériques, enregistrement d'images, etc.).
- Analyse du changement à l'aide de la technique de détection de changement.

Au cours des dix dernières années, diverses méthodes ont été développées, basées sur une unité d'analyse par pixel et par objet. Dans le premier cas, les intensités de pixels sont directement liées au contexte spatial. De nombreuses approches de détection de changement basées sur les pixels ont été proposées pour exploiter les caractéristiques spectrales d'une image : méthodes basées sur une formulation algébrique[19], sur la transformation d'images[14], sur la classification d'images[88] et sur l'apprentissage automatique[11, 86, 9, 13, 41]. Pour les images de haute résolution, on retrouve beaucoup plus d'objets ce qui augmente la diversité des données en entrée. Plus de diversité dit aussi plus de difficulté à traiter l'information qui constitue le contexte de l'image et par le fait même nuit aux performances des approches déjà existantes. Pour les méthodes basées sur la détection d'objets [15] [16][63, 72], les images sont segmentées en objets disjoints et homogènes, ce qui est plus cohérent avec la perception humaine et permet d'analyser les objets au sol en détail en utilisant une abondance d'informations géométriques, spectrales et structurelles.

15. https ://www.kaggle.com/c/noaa-fisheries-steller-sea-lion-population-count/overview
16. https://www.ecodse.org/

Une bonne application de la détection de changement est la détection de véhicules, le suivi et l'estimation des paramètres de trafic[42]. Le problème de la circulation routière est l'un des problèmes urbains majeurs dans les villes fortement peuplées. Avoir des informations plus complètes sur le contrôle, la gestion et la planification du trafic est un besoin vital pour les villes. La surveillance de la circulation permet aux municipalités d'élaborer des programmes de contrôle de la circulation plus précis, de simuler la circulation, d'étudier les îlots de chaleur liés à la densité de véhicules dans le temps et d'étudier plus précisément la relation entre la qualité de l'air et les aérosols liés à la combustion. La vidéo surveillance et les vidéos aériennes/satellites sont deux techniques différentes pour obtenir divers paramètres de trafic.

Les paramètres de trafic peuvent être estimés avec des vidéos aériennes ou satellites. Elles couvrent principalement des tronçons d'autoroute[5], ou une intersection[3], ainsi, une petite zone pourrait être couverte à la fois. De plus, avec une approche aérienne, l'opérateur peut modifier l'état de la mission afin de minimiser les facteurs de perturbation, notamment les ombres et les déplacements de relief. Les articles de Kopsiaftis[44] et al., Mou et al.[62] et Ahmadi et al.[1] sont de bons exemples où des vidéos satellites sont utilisées pour détecter et suivre des objets en mouvement. La connaissance préalable des données SIG (système d'information géographique), à savoir les informations routières et le masque des routes, a été utilisée dans un algorithme de détection comme pour Kopsiaftis[44]. Dans le cas de Yang et al.[90], ils utilisent la carte de chaleur de mouvement et la carte de saillance pour limiter la zone de détection en vue de la détection et de la segmentation en temps réel.

1.3.6 Classification, détection et suivis d'objets

Les méthodes traditionnelles de classification d'images satellites sont discutées dans l'article de Cheng et al.[15]. Ces dernières utilisent des caractéristiques telles que "*Histogram of Ordered Gradients*"(HOG), "*Scale-Invariant Feature Transform*"(SIFT) et les variantes du SIFT, les histogrammes de couleur, etc. Ils discutent également des changements apportés à la télédétection par les méthodes de "*Feature Learning*" non-supervisées, tels que l'analyse en composantes principales, la méthode du "*k-means*", "*sparse coding*", etc.

La classification consiste à étiqueter les régions d'une image. Les méthodes d'apprentissage profond aident énormément pour apprendre les caractéristiques des données elles-mêmes et effectuer une classification aux niveaux les plus avancés.

Images hyperspectrales : La classification est probablement le domaine de recherche le plus actif en analyse de données hyperspectrales. Très souvent, la classification se fait pixel par pixel et les données d'entrée sont converties en un vecteur 1D (bande spectrale) qui correspond normalement à un pixel avec seulement trois valeurs (R-G-B). Il existe une vaste littérature sur des modèles non basés sur les réseaux de neurones conventionnels d'apprentissage automatique[12]. Dans leur article sur l'étude de la classification des images hyperspectrales, Ghamisi et al.[24] ont été constaté que diverses conditions de diffusion atmosphérique compliquaient la diffusion de la lumière. Un des grands avantages avec ce genre de données est que les architectures d'apprentissage profond sont capables d'extraire des informations de haut niveau qui sont généralement plus robustes aux données d'entrée non-linéaires. Le plus grand problème dans ce type de classification est le manque de donnée annotée, souvent aux alentours de 200 bandes spectrales par classes durant l'entraînement.

1.3.7 Représentation 3D et estimation des hauteurs

Il existe plusieurs approches traditionnelles automatiques utilisant des données de télédétection pour construire (ou reconstruire) une géométrie urbaine 3D, parmi lesquelles se trouvent celles basées sur des données de "*Light Detection and Ranging*"(LiDAR). Elles sont les plus largement utilisées pour les aspects 3D du nuage de points[39]. Toutefois, le coût et l'acquisition complexe des données LiDAR limitent leurs applications. Récemment, les méthodes de reconstruction 3D ont été étendues aux images à vues multiples, telles que l'imagerie oblique[82] et la vidéo stéréo[2]. Les modèles 3D peuvent être construits avec précision en comparant et en faisant correspondent les objets avec plusieurs images. Avec des images provenant de caméras verticales et de plusieurs caméras obliques et une carte cadastrale, les bâtiments peuvent être reconstruits avec ses façades 3D et sa structure de toit[27]. Cependant, l'approche qui nous intéresse est celle avec l'utilisation d'images aériennes et satellitaires dans la reconstruction 3D de bâtiments ou retrouver un modèle d'élévation.

1.4 État de l'art des méthodes de segmentation satellitaire multi-classes

Puisque le jeu de données le plus populaire en segmentation pour les images satellitaires est **ISPRS** on va donc prendre les modèles qui performent le mieux sur ce dernier comme état de l'art. Le site web d'**ISPRS** fournit tous les résultats des réseaux qu'ils leur sont envoyés [17] [18]. Cependant, ils ne fournissent que le nom de l'équipe qui a soumis le modèle et fournissent un lien vers un article avec la méthode utilisée seulement si l'équipe leur envoie. Malheureusement, les meilleures méthodes ne sont pas liées à des articles scientifiques et il est donc impossible de fournir un bon état de l'art.

Par chance, un article sorti dernièrement fournit des résultats que les auteurs présentent comme le nouvel état de l'art. Les résultats présentés au tableau 1.5 sont issus de l'article de Diakogiannis et al.[22], ils comparent plusieurs modèles de segmentation sur **Potsdam**. Ce tableau présente les *F1-scores* et la justesse globale des modèles.

Modèles	Routes	Bâtiments	Végétation basses	Arbres	Voitures	*F1-score* moyen	Justesse globale
UZ_1 [84]	89.3	95.4	81.8	80.5	86.5	86.7	85.8
RIT_L7 [54]	91.2	94.6	85.1	85.1	92.8	89.8	88.4
RIT_4 [70]	92.6	97.0	86.9	87.4	95.2	91.8	90.3
DST_5 [78]	92.5	96.4	86.7	88.8	94.7	91.7	90.3
CAS_Y3 [19]	92.2	95.7	87.2	87.6	95.6	91.7	90.1
CASIA2[55]	93.3	97.0	87.7	88.4	96.2	92.5	91.1
DPN_MFFL[66]	92.4	96.4	87.8	88.0	95.7	92.1	90.4
HSN+OI+WBP[56]	91.8	95.7	84.4	79.6	88.3	87.9	89.4
ResUNet-a d7v2 [22]	93.0	97.2	87.5	88.4	96.1	92.4	91.0
ResUNet-a d7v2 [22]	93.5	97.2	88.2	89.2	96.4	92.9	91.5

tableau 1.5: *F1-scores* (en %) par classe de différentes méthodes de segmentation sur **Potsdam**.

17. http://www2.isprs.org/commissions/comm2/wg4/potsdam-2d-semantic-labeling.html
18. http://www2.isprs.org/commissions/comm2/wg4/vaihingen-2d-semantic-labeling-contest.html
19. http://www2.isprs.org/commissions/comm3/wg4/tests.html

À noter que ces méthodes utilisent toutes les modalités qu'**ISPRS** met à leur disposition, "IR-R-G-B-DSM" pour entraîner leurs méthodes et utilise seulement l'ensemble de **Potsdam**.

Chapitre 2

Algorithmes d'apprentissage

Ce chapitre a pour objectif de résumer les notions à la base des algorithmes d'apprentissage, telles que les probabilités, la théorie de l'information, l'apprentissage automatique, ainsi que les réseaux de neurones et l'apprentissage profond.

2.1 Probabilités

Premièrement, à la base des probabilités figure la notion de variable aléatoire. Une **variable aléatoire** est une variable qui prend au hasard des valeurs dans un espace d'échantillonnage. Pour décrire la vraisemblance d'une valeur pouvant être prise par une variable aléatoire x, il faut spécifier une distribution de probabilité. On peut écrire $x \sim P(x)$ pour indiquer que x est une variable aléatoire tirée d'une distribution de probabilité $P(x)$. Les distributions de probabilité sont décrites différemment selon que la variable aléatoire est discrète ou continue. Dans le cas discret, la probabilité est associée à chaque valeur possible, dans le cas continu, la fonction indique la densité de probabilités associée au domaine de la variable.

Les variables aléatoires discrètes sont décrites avec une **fonction de masse**. Une fonction de masse $P(x)$ détermine chaque valeur de l'espace d'échantillonnage de la

variable x. Un exemple simple est la distribution uniforme sur n résultats possibles :

$$P(X = x) = 1/n.$$

Cela signifie : «La probabilité que la variable X prenne la valeur x est de 1 sur le nombre de valeurs possibles».

Il arrive souvent qu'une situation soulève l'intérêt pour la probabilité d'un événement étant donné un autre événement déjà observé. La **distribution de probabilité conditionnelle** de x étant donné y est représentée par :

$$P(x|y) = \frac{P(x, y)}{P(y)}.$$

Où $P(x, y)$ est la probabilité conjointe, une probabilité d'un sous-ensemble de variables aléatoires, dans ce cas x et y. Un bon exemple pour illustrer ce concept est : *le fait de savoir que je porte une veste renseigne sur la météo sans l'observer directement*. Il est important de savoir qu'une distribution conditionnelle est valide uniquement si $P(y) > 0$.

En multipliant les deux côtés de la distribution de probabilités conditionnelles par $P(y)$ apparaît la **règle de probabilité en chaîne** :

$$P(x, y) = P(x|y) \cdot P(y).$$

Note, la règle de probabilité en chaîne pour deux variables peut être écrite de deux manières équivalentes :

$$P(x, y) = P(x|y) \cdot P(y)$$

$$P(x, y) = P(y|x) \cdot P(x)$$

Lorsque l'on utilise la règle de probabilité en chaîne de l'encadrer ci-haut et les notions de la distribution de probabilité conditionnelle, il est possible d'isoler $P(y|x)$ en ramenant $P(x)$ de l'autre côté. Le résultat est ce qu'on appelle **la règle de Bayes** :

$$\begin{aligned} P(x,y) &= P(x,y) \\ P(y|x) \cdot P(x) &= P(x|y) \cdot P(y) \\ P(y|x) &= \frac{P(x|y) \cdot P(y)}{P(x)}. \end{aligned} \tag{2.1}$$

La règle de Bayes est une notion centrale en apprentissage automatique. Cette notion a même inspiré un chapitre complet de l'histoire des mathématiques : *les statistiques bayésiennes*. Cette règle simple permet de mettre à jour les quantités à mesure que nous rassemblons plus d'observations à partir de données.

La valeur attendue, ou l'**espérance mathématique**, d'une fonction $f(x)$ sur une variable aléatoire $x \sim P(x)$ est la valeur moyenne de $f(x)$ pondérée par une distribution $P(x)$. Pour un x discret, l'espérance prend la forme d'une somme :

$$\mathbb{E}_{X \sim P}[f(x)] = \sum_X P(x) \cdot f(x).$$

Dans le cas où x serait continu, on remplacerait la somme par une intégrale. L'espérance agit comme une moyenne pondérée sur $f(x)$, où les poids sont les probabilités de chaque x.

2.2 Théorie de l'information

La théorie de l'information est basée sur la théorie des probabilités et des statistiques et s'articule autour de la quantité d'information présente dans un signal ou une fonction. Tout d'abord, il faut considérer une variable aléatoire X et ce demander combien d'information est reçue lors de son observation. La quantité d'informations peut être considérée comme la surprise associée à un évènement. Moins un évènement est probable et plus l'information qui lui est associée est élevée. Ce principe se

matérialise par l'équation suivante :

$$I_P(x) = -\log P(x)$$

où $I_P(x)$ est l'information qui se produit lors de l'évènement x. Fait intéressant, si la base du log est 2, alors l'information peut être exprimée en "*bit*" et "*nats*" si le logarithme naturel est utilisé.

La quantité moyenne d'informations transmises au cours du processus est obtenue en prenant l'espérance de l'information :

$$H[x] = \mathbb{E}[-\log P(x)] = -\sum_X P(x) \log P(x).$$

Cette quantité est appelée l'**entropie** de la variable aléatoire X.

La divergence de Kullback-Leibler (KL) est une mesure permettant de trouver des similitudes entre deux distributions de probabilité. Elle mesure combien une distribution diverge de l'autre. La divergence KL entre $P(x)$ et $Q(x)$ indique la quantité d'informations perdues lors de l'approximation des données par $P(x)$ avec $Q(x)$. La divergence de KL entre une distribution $Q(x)$ et une distribution $P(x)$ est définie comme suit :

$$D_{KL}(P \parallel Q) = \sum_X P(x) \log \frac{P(x)}{Q(x)}.$$

Il est à noter que la divergence KL n'est pas symétrique et donc que $D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$. De plus, si P et Q sont identiques ($P = Q$), alors $D_{KL}(P \parallel Q) = 0$, et c'est le seul cas où la divergence est symétrique, puisque $D_{KL}(Q \parallel P) = 0$. Avec un peu d'algèbre, la définition de la divergence de KL peut être reformulée en termes d'autres quantités. La formulation la plus utile est :

$$D_{KL}(P \parallel Q) = \mathbb{E}_P \left[\log \frac{P(X)}{Q(X)} \right]$$

$$D_{KL}(P \parallel Q) = \mathbb{E}_P[-\log Q(X)] + \mathbb{E}[\log P(X)]$$

Ici, $\mathbb{E}[-\log Q(X)]$ est l'**entropie croisée** entre P et Q, notée $\mathbb{E}[-\log Q(X)] = H(P, Q)$. Le second terme $\mathbb{E}[\log P(X)] = -H[X]$ est l'entropie de X.

2.3 Apprentissage automatique

Un algorithme d'apprentissage automatique (ou apprentissage machine) est une expression mathématique qui épouse la distribution de données étiquetées. Par exemple, si un détaillant en ligne souhaite anticiper les ventes pour le prochain trimestre, il peut utiliser un algorithme d'apprentissage automatique pour prédire ces ventes en fonction des ventes passées et d'autres données pertinentes.

Le premier scientifique à faire usage de l'expression d'apprentissage automatique a été Arthur Samuel suite de la création de son programme pour IBM en 1952 [1]. Le programme jouait au Jeu de Dames et s'améliorait en jouant. En d'autres mots, un programme informatique capable d'apprendre à partir d'expériences pour améliorer ses performances sur une tâche donnée.

2.3.1 Comment s'effectue l'apprentissage

L'apprentissage est un processus ayant pour but d'apprendre les paramètres d'une fonction de prédiction afin de réduire au mieux une fonction de coût (R). Les algorithmes d'apprentissage automatique peuvent prendre plusieurs formes et peuvent être généralement catégorisés en deux catégories : les algorithmes supervisés et non supervisés. Cette distinction exprime la manière dont les algorithmes utilisent les données lors de l'entraînement.

Algorithmes non supervisés

L'apprentissage non supervisé consiste à disposer que de données d'entrée ($\vec{x} \in \mathbb{R}^d$) et aucune variable cible correspondante. L'objectif de l'apprentissage non supervisé est de modéliser la distribution sous-jacente des données. Voici des exemples populaires d'algorithmes d'apprentissage non supervisé :

1. http://infolab.stanford.edu/pub/voy/museum/samuel.html

- Apprentissage de distribution $P(\vec{x})$.
- "*k-means*" pour les problèmes de "*clustering*".
- Algorithme a priori pour les problèmes d'apprentissage d'association.
- Réduction de dimensionnalité.

Algorithmes supervisés

À la base des algorithmes d'apprentissage supervisé figure un ensemble de données consistant en des variables d'entrée ($\vec{x}$), une variable cible (y) et un ensemble de paramètres de la fonction ($\vec{w}$). Dans ce cas-ci, un algorithme doit apprendre une fonction de mappage de l'entrée à la sortie :

$$y = f_{\vec{w}}(\vec{x}).$$

Le but est d'apprendre si bien la fonction de mappage que lorsque la fonction est appliquée à de nouvelles données d'entrée ($\vec{x}$), la prédiction des variables de sortie (y) pour ces données reste être performante. C'est ce qu'on appelle l'apprentissage supervisé, puisqu'un algorithme passe par l'ensemble de données étiquetées lors de l'entraînement, ces données peuvent être considérées comme des enseignants supervisant le processus d'apprentissage du réseau. Le terme supervisé vient principalement de la connaissance des cibles.

Quelques exemples populaires d'algorithmes d'apprentissage automatique supervisé :

- "*Random forest*" pour les problèmes de classification et de régression.
- "*Support vector machines*"(SVM) pour les problèmes de classification.
- Réseaux de neurones.

Dans le cas des algorithmes supervisés, la fonction de coût est écrite :

$$R = \mathbb{E}_{\vec{x},y\sim p_{data}}\left[\mathcal{L}(f_{\vec{w}}(\vec{x}), y)\right] = \int \mathcal{L}(f_{\vec{w}}(\vec{x}), y) dP(\vec{x}, y) \tag{2.2}$$

où cela revient à calculer l'espérance mathématique($\mathbb{E}$) de la fonction de perte($\mathcal{L}$). Cette quantité est connue aussi sous le nom de *risque*(R). Puisque l'objectif est de

minimiser le *risque*, ce problème devient un problème d'optimisation. En appliquant une approximation *Monte-Carlo* au *risque*, on obtient une approximation pour l'espérance mathématique avec une distribution empirique $\hat{p}(x, y)$. Ce qui nous mène au le risque empirique[83] :

$$\mathbb{E}_{\vec{x},y\sim\hat{p}(\vec{x},y)}[\mathcal{L}] \approx \frac{1}{m}\sum_{i}^{m}\mathcal{L}(f_{\vec{w}}(\vec{x}_i), y_i) \tag{2.3}$$

avec m le nombre d'éléments pour l'entraînement. Lorsque l'on minimisera le risque empirique en fonction de l'ensemble des paramètres de f on se retrouve avec :

$$\arg\min_{\vec{w}} \mathbb{E}[\mathcal{L}] = \arg\min_{\vec{w}} \frac{1}{m}\sum_{i}^{m}\mathcal{L}(f_{\vec{w}}(\vec{x}_i), y_i). \tag{2.4}$$

Cette forme d'optimisation sera mieux expliquée dans la partie de l'erreur absolue moyenne et l'erreur quadratique moyenne.

2.3.2 Maximum de vraisemblance et a *posteriori*

Métaphoriquement, l'étude que l'algorithme effectue sur la tâche peut être associée au **maximum de vraisemblance**. On cherche à trouver les paramètres $\vec{w}$ du modèle via un ensemble de données étiquetées connues $\{X, Y\}$. La distribution de vraisemblance est écrite $P(Y, X|\vec{w})$. Puisque l'on cherche à maximiser cette distribution et par les propriétés énoncées à la section 2.1, cela revient à maximiser $P(Y|X, \vec{w})$. Comme le nom l'indique, le but est de trouver les paramètres $\vec{w}$ qui maximisent cette distribution,

$$w^*_{MV} = \arg\max_{\vec{w}} P_{model}(Y|X, \vec{w})$$

On assument l'indépendance des données.

$$w^*_{MV} = \arg\max_{\vec{w}} \prod_{i=1}^{n} P(\vec{y}_i|\vec{x}_i, \vec{w})$$

Prendre un produit de nombres inférieurs à 1 s'approcherait dangereusement de 0, puisque le nombre d'éléments n peut être arbitrairement grand. Il n'est alors pas pratique de calculer le produit de ces probabilités. Néanmoins, travailler dans l'espace des *logs* peut s'avérer intéressant, car le logarithme augmente de façon monotone.

$$w^*_{MV} = \arg\max_{\vec{w}} \sum_{i=1}^{n} \log P(\vec{y}_i|\vec{x}_i, \vec{w})$$

Toutefois, notons qu'il existe un estimateur similaire à celui du **maximum de vraisemblance**, appelé **maximum *a posteriori*** (MAP). Ce qui les différencie est que le **maximum *a posteriori*** permet d'optimiser la vraisemblance de $\vec{w}$ par rapport à X et la probabilité des étiquettes Y, tout en considérant une distribution *a priori* sur $\vec{w}$.

$$\begin{aligned} w^*_{MAP} &= \arg\max_{\vec{w}} P_{modele}(\vec{w}|X, Y) \\ &= \arg\max_{\vec{w}} \frac{P(Y|\vec{w}; X)P(\vec{w}; X)}{P(Y)} \qquad \text{(règle de Bayes)} \\ &\propto \arg\max_{\vec{w}} \prod_{i=1}^{n} P(\vec{y}_i|\vec{x}_i; \vec{w}) \cdot P(\vec{w}). \end{aligned}$$

Pour la même raison que le **maximum de vraisemblance**, un log est utilisé pour convertir le produit de probabilités en une somme de log.

$$\begin{aligned} w^*_{MAP} &= \arg\max_{\vec{w}} \ \log\left(\prod_{i=1}^{n} P(\vec{y}_i|\vec{x}_i; \vec{w}) \cdot P(\vec{w})\right) \\ &= \arg\max_{\vec{w}} \ \log \prod_{i=1}^{n} P(\vec{y}_i|\vec{x}_i; \vec{w}) + \log P(\vec{w}) \\ &= \arg\max_{\vec{w}} \sum_{i=1}^{n} \log P(\vec{y}_i|\vec{x}_i; \vec{w}) + \log P(\vec{w}). \end{aligned}$$

En comparant les équations du **maximum de vraisemblance** et le **maximum *a posteriori***, la seule différence est l'inclusion du modèle *a priori* $P(\vec{w})$ dans le **maximum *a posteriori***, autrement ces équations sont identiques. Cela signifie que la probabilité est maintenant pondérée par les connaissances *a priori* que nous avons obtenues des paramètres $\vec{w}$.

2.3.3 Métriques d'évaluation

Un modèle peut donner des résultats satisfaisants lorsqu'évalué à l'aide d'une métrique, mais peut donner de mauvais résultats lorsqu'évalué par rapport à d'autres mesures. Les méthodes d'évaluation sont aussi appelées *métriques*. La méthode la plus communément utilisée en classification est la justesse, mais avant, voyons les notions de base avec la matrice de confusion.

Matrice de confusion

La matrice de confusion, comme son nom l'indique, donne une matrice en sortie et décrit les performances complètes du modèle en fonction des différentes valeurs possibles en prédiction. La matrice permet d'illustrer le nombre de prédictions correctes et incorrectes par classe. La matrice de confusion montre les façons dont le modèle classifie les images et à quel point il est confus quand il fait des prédictions. Cela donne un aperçu non seulement des erreurs commises par le classificateur, mais surtout des types d'erreurs qui sont commises et si le classificateur a de la difficulté à distinguer deux classes en particulier.

Supposons un problème de classification binaire. Les échantillons appartiennent à deux classes : positif ou négatif. Le classificateur prédit une classe pour un échantillon d'entrée donné. En testant le modèle sur 165 échantillons, les résultats sont les suivants :

		Prédit	
	N=165	positif	négatif
Vérité	positif	50	10
	négatif	5	100

On retrouve les quatre attributs suivants :

- Vrais positifs : Les cas dans lesquels ***positif*** est prédit et la cible était également ***positif***.
- Vrais négatifs : les cas dans lesquels ***négatif*** est prédit et la cible était ***négatif***.
- Faux positifs : les cas dans lesquels ***positif*** est prédit et la cible était ***négatif***.
- Faux négatifs : Les cas dans lesquels ***négatif*** est prédit et la cible était ***positif***.

La précision de la matrice est calculée en prenant l'équation 2.5.

Justesse de la classification

La justesse (le terme anglais est "*accuracy*") est le rapport entre le nombre de prédictions correctes et le nombre total d'échantillons évalués.

$$Justesse = \frac{Nombre\ de\ bonnes\ prédictions}{Nombre\ totale\ de\ prédictions}.$$

Cette métrique fonctionne à son meilleur lorsqu'il y a un nombre égal d'échantillons appartenant à chaque classe.

Par exemple, pour évaluer les algorithmes de classification 2 classes, on combine les attributs suivants : vrais positifs (TP), faux positifs (FP), vrais négatifs (TN) et faux négatifs (FN).

$$Justesse = \frac{TP + TN}{TP + TN + FP + FN}$$

La justesse est toutefois non recommandée pour des données fortement déséquilibrées. Par exemple, lorsqu'il y a 98% d'échantillons d'une classe **A** et 2% d'échantillons d'une classe **B** dans l'ensemble de formation. Ensuite, le modèle peut facilement obtenir une précision d'entraînement de 98% en prédisant simplement chaque échantillon d'apprentissage appartenant à la classe **A**. Lorsque le même modèle est testé sur un ensemble de test avec 60% d'échantillons d'une classe **A** et 40% d'échantillons d'une classe **B**, la précision du test tombe alors à 60%. La précision de la classification est excellente, mais elle peut donner un faux sentiment d'atteindre de bonnes performances. Le vrai problème se pose lorsque le coût d'une mauvaise classification des échantillons de classe mineure est très élevé. Si le réseau a pour objectif de traiter une

maladie rare, mais mortelle, le coût de l'échec du diagnostic de la maladie d'une personne malade est beaucoup plus élevé que le coût de l'envoi d'un mauvais diagnostic à une personne en bonne santé.

F-Measure

Pour nuancer l'analyse, on introduit les métriques de précision et de rappel. La précision est élevée lorsque le nombre de vrais positifs parmi tous les positifs prédits est élevé.

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

La précision accorde plus d'importance aux faux positifs, peu importe le nombre de faux négatifs. À l'opposé, le rappel est élevé si le nombre de faux négatifs est faible et le nombre de vrais positifs est élevé. Il est important que tous les éléments de la classe positive soient identifiés même si cela génère un grand nombre de faux positifs.

$$Rappel = \frac{TP}{TP + FN}$$

Ces deux métriques ont des objectifs très différents et étant donné leurs objectifs opposés, il est dangereux de n'utiliser qu'une seule des deux. C'est pour cela que la *F-measure* est plus utilisée. Elle est une combinaison pondérée de la précision et du rappel :

$$F_\beta = (1 + \beta^2)\frac{\text{Précision} \cdot \text{Rappel}}{\beta^2\text{Précision} + \text{Rappel}}$$

Le paramètre β permet de mettre l'accent sur la précision ou le rappel. En général $\beta = 1$.

Le F1 Score est la moyenne harmonique entre la précision et le rappel. La plage du score F1 est [0, 1] et est ensuite multipliée par 100 pour affiché un résultat en pourcentage. Ce résultat indique la précision du classificateur (combien d'instances il classe correctement), ainsi que sa robustesse (savoir s'il ne manque pas un nombre significatif d'instances). Cette métrique offre une précision extrême lorsque le modèle a une haute précision, mais un rappel moindre. Dans cette situation, il manque alors

un grand nombre d'instances difficiles à classer. Plus le score F1 est élevé, meilleures sont les performances du modèle. Mathématiquement, il peut s'exprimer comme suit :

$$F1 = 2 \cdot \frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Erreur absolue moyenne

L'erreur absolue moyenne ("*Mean Absolute Error*" ou **MAE** en anglais) est la moyenne de la différence entre les valeurs d'origine et les valeurs prédites par le modèle. Le calcul donne la mesure de la distance entre les prévisions et la sortie réelle. Cependant, le résultat ne donne aucune idée si le modèle sous-prédit ou sur-prédit sur l'ensemble de données. Mathématiquement, l'erreur absolue moyenne est représentée comme :

$$\text{Erreur Absolue Moyenne} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - y_i^{pred}\right|.$$

Erreur quadratique moyenne

L'erreur quadratique moyenne (*Mean Squared Error* ou **MSE** en anglais) est similaire à l'erreur absolue moyenne, la seule différence étant que l'erreur quadratique moyenne prend la moyenne différence au carré entre les valeurs d'origine et les valeurs prédites du modèle. L'avantage de l'erreur quadratique moyenne est qu'il est plus facile de calculer le gradient et que l'erreur quadratique est associée à un bruit Gaussien :

$$y = f(x) + \mathcal{N}(0, \sigma^2)$$

Puisque ce calcul prend en considération le carré de l'erreur, l'effet des erreurs plus importantes devient encore plus prononcé que l'erreur plus petite. Le modèle peut alors se concentrer davantage sur les erreurs plus importantes.

$$\text{Erreur quadratique moyenne} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - y_i^{pred}\right)^2.$$

2.4 Réseaux de neurones et l'apprentissage profond

L'apprentissage profond regroupe les algorithmes utilisant les réseaux de neurones pour effectuer leur tâche. La définition d'un neurone est écrite comme une fonction d'activation h :

$$h(\langle \vec{w}, \vec{x} \rangle + b)$$

où $\vec{x}$ sont les données en entrée, $\vec{w}$ représente les poids (sous forme de vecteur) appris et b est le biais qui est appris au même moment. La fonction non-linéaire h est appliquée au résultat pour que le neurone puisse approximer des fonctions non linéaires.

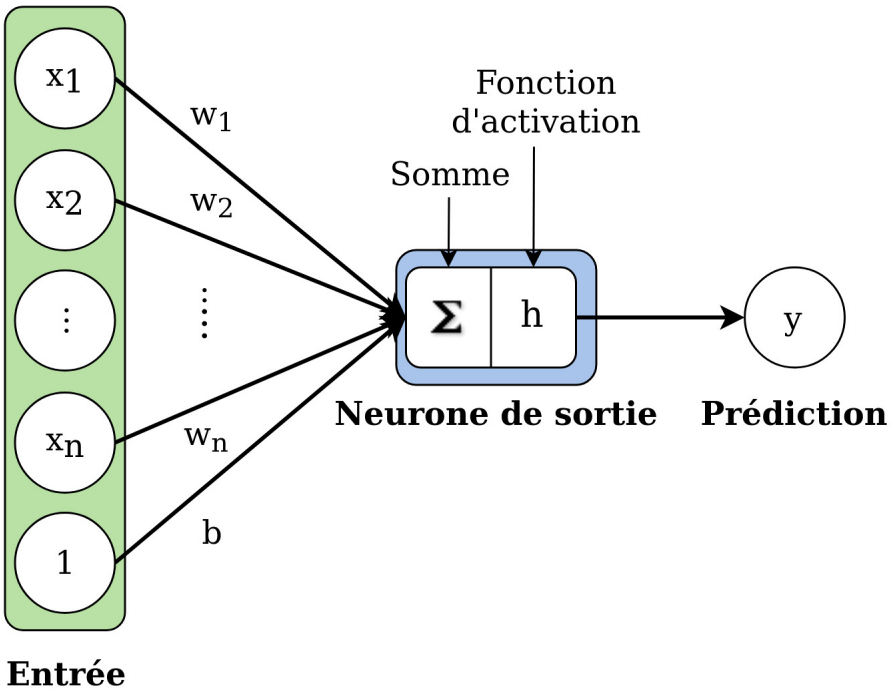


figure 2.1: Figure illustrant un neurone.

La forme la plus populaire pour représenter les réseaux de neurones est sous forme de graphe orienté où des arrêtes font le pont entre les neurones comme à la figure 2.1. Pour simplifier la notation, on inclut le biais b dans les poids $\vec{w}$ et ajoute une

dimension au vecteur $\vec{x}$:

$$x \to \vec{x}' = (\vec{x}, 1),$$

$$w \to \vec{w}' = (\vec{w}, b).$$

Alors, un neurone est simplement un produit scalaire combiné à une fonction d'activation non linéaire.

2.4.1 Perceptron

Le premier réseau de neurones jamais publié fut le perceptron [74]. Défini par :

$$y = h(\langle \vec{w}, \vec{x} \rangle)$$

où h est la fonction de non-linéarité signe. Pour la classification binaire le perceptron prend la forme suivante :

$$y = h(\langle \vec{w}, \vec{x} \rangle) = \begin{cases} -1 & , \text{si } \langle \vec{w}, \vec{x} \rangle < 0 \\ +1 & , \text{si } \langle \vec{w}, \vec{x} \rangle \geq 0 \end{cases}.$$

Pour ce modèle, il a fallu une fonction de perte $\mathcal{L}$ spécifique appelée le critère du perceptron représente par,

$$\mathcal{L}_{perceptron}(\mathcal{M}; \vec{w}) = - \sum_{(\vec{x}_i, y_i) \in \mathcal{M}} y_i \langle \vec{w}, \vec{x}_i \rangle.$$

Avec $\mathcal{M}$ représentant les exemples mal-classés.

On peut maintenant parler de la descente de gradient qui permet au modèle d'apprendre les poids ($\vec{w} \leq \vec{w} + \nabla L$). On peut calculer le gradient de la fonction de perte comme suit :

$$\nabla_{\vec{w}} \mathcal{L}_{perceptron} = - \sum_{(\vec{x}_i, y_i) \in \mathcal{M}} y_i \vec{x}_i.$$

Lorsque toutes les données sont bien classées on obtient un gradient nul $\nabla \mathcal{L} = 0$ et cela même si la frontière de décision apprise est collée aux données. Alors, lorsque

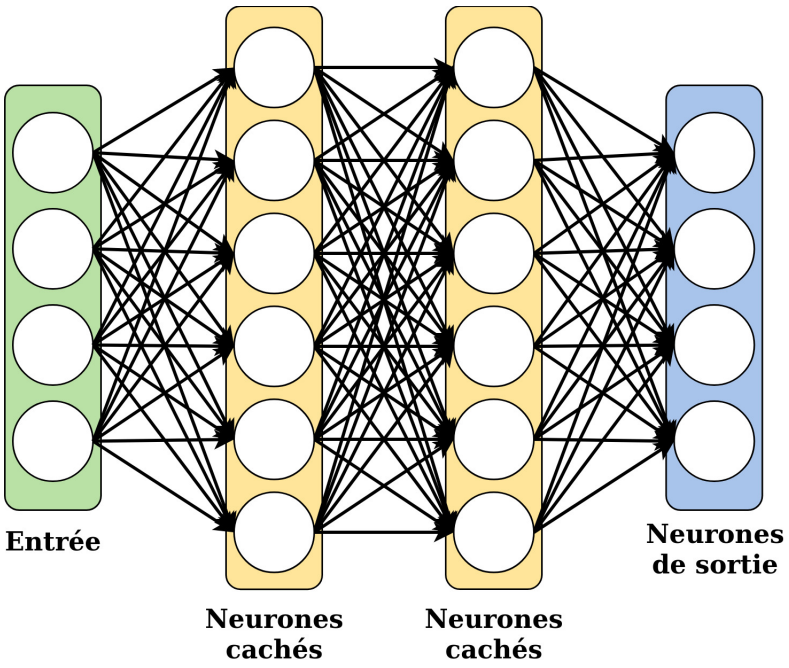


figure 2.2: Exemple de Perceptron[74] multi-couches et multiclasses.

les données sont très proches de la distribution de probabilité, le modèle va bien performer sur les mêmes données qui ont été données en entrée. Cependant, si de nouvelles données sont données en entré s'écarte de la distribution, le réseau ne sera pas en mesure de bien généraliser et performera moins bien. Cela est un problème, car on veut que le modèle puisse généraliser sur des contextes différents afin d'avoir une surface de séparation qui sépare mieux les données, on utilise la **régression logistique**. À la place de prédire la classe y_i à partir de la donnée d'entrée $\vec{x}_i$, on calcule $P(y_i|\vec{x}_i;\vec{w})$. On prédit la fonction **sigmoïde** $\sigma(a) = \frac{1}{1+e^{-a}}$.

On obtient la sigmoïde en appliquant le théorème de Bayes 2.1 à la probabilité conditionnelle de la classe $C1$.

$$\begin{aligned} p(C_1|\vec{x}_i) &= \frac{p(\vec{x}_i|C_1)p(C_1)}{P(\vec{x}_i|C_1)p(C_1) + p(\vec{x}_i|C_2)p(C_2)} \\ &= \frac{1}{1 + \frac{p(\vec{x}_i|C_2)p(C_2)}{p(\vec{x}_i|C_1)p(C_1)}} \\ &= \frac{1}{1 + e^{-a}} \\ &= \sigma(a) \end{aligned}$$

où on définie a comme :

$$a = \ln \frac{p(\vec{x}_i)|C_1)p(C_1)}{p(\vec{x}_i|C_2)p(C_2)}$$

La fonction de perte d'un réseau logistique est ce qu'on appelle **entropie croisée** dont sa formulation se dérive de la distribution de Bernouilli :

$$\begin{aligned} \mathcal{L} &= \textstyle\sum_i y_i \log(\sigma(\langle \vec{w}, \vec{x}_i \rangle)) + (1 - y_i) \log(1 - \sigma(\langle \vec{w}, \vec{x}_i \rangle)) \\ \nabla_{\vec{w}} \mathcal{L} &= \textstyle\sum_i (y_i - \sigma(\langle \vec{w}, \vec{x}_i \rangle)) \vec{x}_i. \end{aligned}$$

2.4.2 *Softmax*

On généralise la régression logistique au cas multiclasses à l'aide d'une fonction appelée ***softmax***. Il faut premièrement modifier le réseau pour que la dernière couche contienne autant de neurones que de classes (figure 2.2) et pour simplifier la notation mathématique, on va seulement considérer un réseau sans couche cachée. Sous la forme mathématique, cela revient à substituer le vecteur de poids ($\vec{w}$) par une matrice $\mathbf{W} = \{\vec{w}_1, \ldots, \vec{w}_K\}$ où K est le nombre de classes. En deuxième lieu, le produit matriciel $\mathbf{W}\vec{x}$ prend la place du produit scalaire. Finalement, la fonction sigmoïde est remplacée par la fonction *softmax* :

$$\begin{gathered} \sigma(\langle \vec{w}, \vec{x}_i \rangle) \rightarrow softmax(\mathbf{W}\vec{x}_i) \\ f^k_{\vec{w}}(\vec{x}_i) = softmax(\mathbf{W}\vec{x}_i) = \frac{e^{\vec{w}_k \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}} \end{gathered}$$

En sortie, on obtient un vecteur à K dimensions où chaque élément représente la

probabilité que x_i appartienne à la classe $y_k = P(y_k|x_i)$. Au moment de la prédiction, on prendra la probabilité la plus élevée pour trouver la classe la plus probable.

$$y_k = \arg\max_k \frac{e^{\vec{w}_k \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}}$$

On peut alors définir le gradient pour chaque sortie du réseaux par :

$$\frac{\partial\ f_{\vec{w}}^k(\vec{x}_i)}{\partial \vec{w}_l} = \frac{\partial \left(\frac{e^{\vec{w}_k \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}} \right)}{\partial \vec{w}_l}$$

Lors de la dérivée, on va retrouver deux cas possibles, $k = l$ et $k \neq l$. Pour $k = l$ on se retrouve avec :

$$\begin{aligned}
\frac{\partial\ f_{\vec{w}}^k(\vec{x}_i)}{\partial \vec{w}_l} &= \frac{\partial \left(\frac{e^{\vec{w}_k \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}} \right)}{\partial \vec{w}_l} \\
&= \vec{x}_i\ \frac{\left(e^{\vec{w}_k \vec{x}_i} \sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i} \right) - \left(e^{\vec{w}_k \vec{x}_i} e^{\vec{w}_l \vec{x}_i} \right)}{\left(\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i} \right)^2} \\
&= \vec{x}_i\ \frac{e^{\vec{w}_k \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}}\ \frac{\left(\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i} \right) - e^{\vec{w}_l \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}} \\
&= \vec{x}_i\ \frac{e^{\vec{w}_k \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}}\ \left(1 - \frac{e^{\vec{w}_l \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}} \right) \\
&= \vec{x}_i\ f_{\vec{w}}^k(\vec{x}_i)\ \left(1 -\ f_{\vec{w}}^k(\vec{x}_i) \right)
\end{aligned}$$

Pour le cas $k \neq l$:

$$\frac{\partial\ f_{\vec{w}}^k(\vec{x}_i)}{\partial \vec{w}_l} \quad = -\vec{x}_i\ \frac{e^{\vec{w}_l \vec{x}_i} e^{\vec{w}_k \vec{x}_i}}{\left(\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i} \right)^2}.$$

Puisque l'on utilise l'entropie croisée comme fonction de perte $\mathcal{L}$, la fonction de perte devient :

$$\mathcal{L}(\mathcal{M}, \mathbf{W}) = -\sum_{i}^{n} \sum_{k}^{K} \vec{y}_i \log \left(\frac{e^{\vec{w}_k \vec{x}_i}}{\sum_{j=1}^{K} e^{\vec{w}_j \vec{x}_i}} \right)$$

On peut réécrire son gradient comme suit :

$$
\begin{aligned}
\nabla_{\vec{w}_l}\mathcal{L} &= -\sum_{i}^{n}\sum_{k}^{K}\vec{y}_{i,k}\frac{\partial \log\left(softmax(\mathbf{W}\vec{x}_i)\right)}{\partial \vec{w}_l} \\
&= -\sum_{i}^{n}\sum_{k}^{K}\vec{y}_{i,k}\ \frac{1}{softmax(\mathbf{W}\vec{x}_i)}\ \frac{\partial\ softmax(\mathbf{W}\vec{x}_i)}{\partial \vec{w}_l} \\
&= -\sum_{i}^{n}\left(\sum_{k}^{K}\vec{y}_{i,k}\left(\vec{x}_i\ \left(1-\frac{e^{\vec{w}_l\vec{x}_i}}{\sum_{j=1}^{K}e^{\vec{w}_j\vec{x}_i}}\right)\right)-\sum_{k\neq l}\vec{y}_{i,k}\left(-\vec{x}_i\ \frac{e^{\vec{w}_l\vec{x}_i}}{\sum_{j=1}^{K}e^{\vec{w}_j\vec{x}_i}}\right)\right) \\
&= -\sum_{i}^{n}\vec{x}_i\left(\sum_{k}^{K}\left(\vec{y}_{i,k}-\frac{\vec{y}_{i,k}\ e^{\vec{w}_l\vec{x}_i}}{\sum_{j=1}^{K}e^{\vec{w}_j\vec{x}_i}}\right)+\sum_{k\neq l}\frac{\vec{y}_{i,k}\ e^{\vec{w}_l\vec{x}_i}}{\sum_{j=1}^{K}e^{\vec{w}_j\vec{x}_i}}\right) \\
&= \sum_{i}^{n}\vec{x}_i\left(\frac{e^{\vec{w}_l\vec{x}_i}}{\sum_{j=1}^{K}e^{\vec{w}_j\vec{x}_i}}-\vec{y}_{i,l}\right)
\end{aligned}
$$

Comme dans la figure 2.2, il est possible d'ajouter plusieurs couches cachées au sein d'un perceptron pour approximer des fonctions non linéaires et et ainsi en augmenter la capacité. Pour le cas multiclasses avec un réseau à N couches, le *softmax* se réécris :

$$f_{\mathbf{W}}(\vec{x}_i) = softmax\ (\mathbf{W}^N h^{N-1}(\dots h^1(\mathbf{W}^1\vec{x}_i))\dots)$$

où W^n représente une matrice de poids pour la couche n et h^n est la fonction d'activation à la n^{eme} couche.

2.4.3 Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs, ou *Convolutional Neural Network* en anglais, ont sans doute l'architecture d'apprentissage profond la plus populaire. L'intérêt porté aux CNN est principalement dû au fait que c'est le modèle idéal pour tous les problèmes liés a l'analyse d'images. Il utilise des opérations de convolution, permettant le partage de paramètres.

Couche convolutive

L'arme secrète des CNN est une couche de neurones appelée **couche convolutive**. Dans le cas des images, il s'agit d'une convolution 2D qui consiste à faire glisser un

filtre pour calculer le degré de correspondance du motif à chaque position dans l'image. La taille du noyau est variable et laissée à la discrétion de l'architecte du réseau, le plus commun étant 3×3. Le résultat obtenu à la suite de la convolution du filtre avec une image est appelé carte d'activation, ou *feature map* en anglais. Voici l'équation d'une convolution 2D (S) sur une entrée (I) et un noyau (K) de taille $M \times N$:

$$S(i,j) = \sum_{m,n} K(m,n) I(i-m, j-n) \tag{2.6}$$

Mathématiquement parlant, une opération de convolution effectuée sur deux tenseurs de différentes grosseurs résulte en un changement de dimension de la matrice de sortie. Dans le cas d'une image couleur, l'image d'entrée aurait une dimension de $n_{entree} \times m_{entree} \times canaux$, où "$canaux$" correspond aux couleurs ($RGB \rightarrow 3$). La convolution s'effectuerait avec un filtre de dimension $largeur \times hauteur \times canaux$, comme par exemple, $3 \times 3 \times 3$. Le résultat de l'opération sera alors une image $n_{sortie} \times m_{sortie} \times 1$. Les relations entre $\{n_{entree}, n_{sortie}\}$ et $\{m_{entree}, m_{sortie}\}$ sont :

$$n_{sortie} = \left[\frac{n_{entree} + 2p - k}{s}\right] + 1, \qquad m_{sortie} = \left[\frac{m_{entree} + 2p - k}{s}\right] + 1$$

où p est le "*padding*" ajouté à l'image d'entrée et s le "*stride*" utilisé durant la convolution.

Couche convolutive transposée

Le but : reconstruire l'image d'entrée. Cette couche est décrite comme l'inverse de la couche convolutive. Ces couches combinées peuvent être interprétées comme les passes avant et arrière d'un réseau. Le besoin de convolutions transposées découle généralement du désir d'utiliser une transformation allant dans le sens opposé d'une convolution normale. Par exemple, on pourrait utiliser une telle transformation comme couche de décodage d'un autoencodeur convolutif ou pour projeter des cartes d'entités dans un espace de dimension supérieure.

Contrairement à la théorie où l'on doit transposer le noyau avant de l'appliquer, les bibliothèques d'apprentissage profond ne font pas cette opération, pour la plupart.

Cette opération est appelée la **cross-corrélation**. Par abus de langage, on appelle tout de même cette opération une convolution dans ce document. Une propriété intéressante de transposer le noyau est que la convolution est commutative :

$$S(i,j) = \sum_{m,n} I(m,n)K(i-m, j-n).$$

Pour des questions de rapidité, l'équation 2.6 est utilisée au lieu de cette dernière.

Sous-échantillonnage

L'opération de sous-échantillonnage (*Pooling* en anglais) consiste à réduire la taille des données afin de réduire le nombre de paramètres dans le réseau.

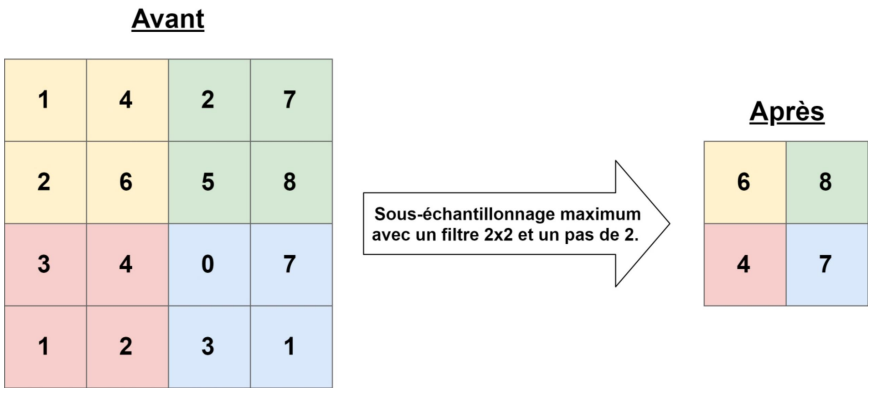


figure 2.3: Résultat d'un exemple d'un sous-échantillonnage maximum. Dans cet exemple, la taille du noyau est de 2×2 et le pas de glissement est de 2.

Par exemple, à partir de chaque fenêtre de taille 2×2 des données d'entrée, la valeur maximale du pixel est sélectionnée et un nouveau bloc de données est ainsi obtenu (voir la figure 2.3). La taille du filtre et des pas sont deux hyper-paramètres importants dans le cadre de cette opération.

L'idée dernière cette opération est de ne conserver que les caractéristiques importantes (neurones avec la valeur maximale) dans chacune des régions et de faire fît des informations qui ne sont pas importantes. On sous-entend par cela que les informations extraites des valeurs maximales décrivent le mieux le contexte de l'image.

Un point très important à noter ici est que l'opération de convolution et spécialement l'opération de sous-échantillonnage réduit la taille de l'image. C'est ce qu'on appelle un échantillonnage à la baisse (*down sampling* en anglais). L'exemple de la figure 2.3, la taille de l'image avant le sous-échantillonnage est 4×4 et après le sous-échantillonnage est 2×2. En fait, le sous-échantillonnage signifie essentiellement la conversion d'une image haute résolution en une image basse résolution.

Maintenant, lors de l'opération de convolution, les filtres de la couche suivante pourront voir un contexte plus large. En d'autres mots, le réseau devient plus profond, la taille des cartes d'activation diminue, mais le champ récepteur augmente.

Suréchantillonnage

Le suréchantillonnage, n'est rien d'autre que l'objectif inverse de celui du sous-échantillonnage, soit augmenté le nombre de dimensions de l'image. Cela peut être utilisé dans plusieurs cas comme celui des GAN (*Generative Adversarial Network*) où l'intention est de construire une image à partir d'un échantillon vectoriel aléatoire imitant une image à partir de la réalité ou de la distribution réelle. Il y en a beaucoup d'autres applications comme l'amélioration de la qualité de l'image, etc.

Lors du sous-échantillonnage, notre intention était assez simple et claire, mais avec le suréchantillonnage, ce n'est pas si simple. Il faut en quelque sorte augmenter les dimensions de l'image et combler les lacunes laissées par cette augmentation.

2.4.4 Réseaux à convolution dédiés à la segmentation d'images

La combinaison de plusieurs couches convolutive et de sous-échantillonnage mène à des réseaux de classification d'image comme le AlexNet[45], VGG[79], ResNet[30] et DenseNet[34]. Si les couches de sous-échantillonnage réduisent les dimensions de hauteur et de largeur, mais si on réduit les dimensions, on peut aussi les augmenter à la même taille que l'image d'origine avec les couches de suréchantillonnage. Lorsque l'on combine les couches de suréchantillonnage et des couches convolutive transposées à la fin d'un réseau de classification, on obtient une classification par pixel. Aussi

appelés des réseaux de segmentation d'image, en voici quelques-uns utiliser au cours des expériences.

Conv-Deconv

Le premier (et le plus simple) des réseaux de segmentation est le *conv-deconv*[64]. Comme le montre la figure 2.4, ce CNN est composé de deux parties : un encodeur et un décodeur. L'encodeur est illustré par les 13 premiers blocs de la figure 2.4. Il fonctionne comme un extracteur de caractéristiques qui transforme l'image $N \times N$ d'entrée en un espace de caractéristiques multidimensionnelles (représentée par la boîte bleu foncé sur la figure 2.4). Dans notre implémentation, l'encodeur a la même architecture que VGG16 [79] mais sans la couche de classification. L'encodeur a 13 couches convolutives et utilise parfois des opérations de rectification et de mise en commun.

Quant au décodeur, il prend la représentation de haut niveau de l'image et la "*mappe*" sur une carte de segmentation à travers une série de convolutions et de couches non linéaires (cf. les 13 dernières cases de la figure 2.4). Le décodeur est une version miroir de l'encodeur, mais avec des opérations de suréchantillonnage (également appelées opérations de "déconvolution") au lieu de regroupements. Les opérations de déconvolution les plus simples sont un suréchantillonnage du voisin le plus proche, soit un suréchantillonnage bilinéaire. Dans notre cas, nous avons utilisé une couche de convolution transposée [2]. La couche utilise une transformation allant dans le sens inverse d'une convolution normale tout en conservant un motif de connectivité compatible avec la dite convolution.

La sortie du softmax est une carte de probabilité $N \times N \times K$ indiquant la probabilité que chaque pixel appartienne à chacune des K classes prédéfinies. Comme mentionné précédemment, $K = 6$ classes pour les jeux de données ISPRS.

2. https://keras.io/layers/convolutional/conv2dtranspose

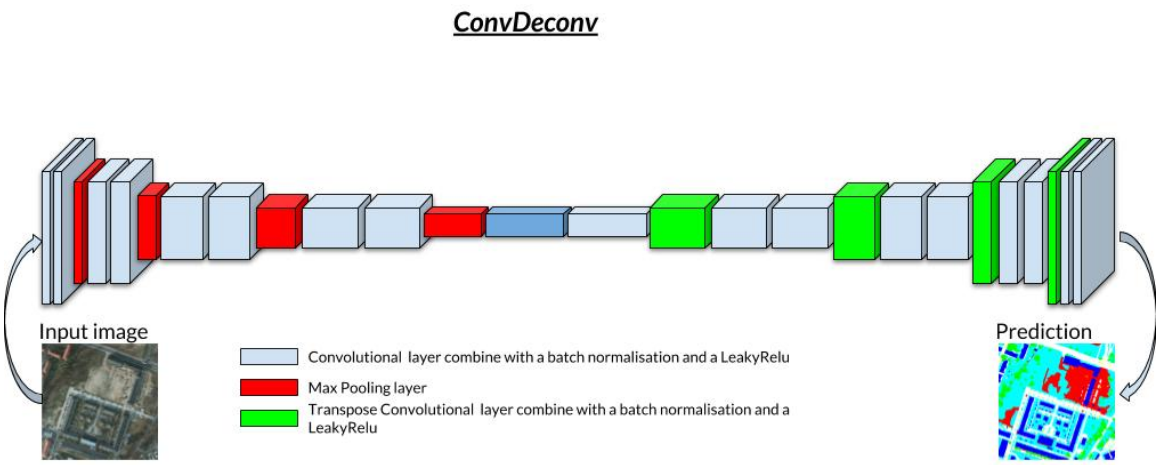


figure 2.4: Le modèle *conv-deconv* [64].

Unet

Le réseau UNet [73] est sans doute le réseau encodeur-décodeur le plus fréquemment utilisé pour la segmentation sémantique. Bien qu'il ait été initialement conçu pour la segmentation d'images biomédicales, il s'est avéré efficace pour diverses applications de segmentation sémantique [38].

Bien que la structure du réseau U-Net soit très similaire à celle du *conv-deconv*, il est toutefois muni de *skip connections* ou saut de connexion. Sans ces sauts de connexion, les contours des cartes de segmentation générés par le décodeur pourraient être inexacts ou moins bien définis, mais sans ces sauts, le UNet [73] est identique au *conv-deconv*[64]. Ils permettent aux sorties intermédiaires de l'encodeur d'être concaténées avec les entrées des couches intermédiaires du décodeur à des positions appropriées pour que les tailles des couches correspondent. Les sauts de connexions des couches précédentes fournissent les informations nécessaires aux couches du décodeur pour créer des contours plus précis et par le fait même faire de meilleures prédictions.

Comme le montre la figure 2.5, un saut de connexion concatène les couches de caractéristiques renvoyées par un bloc encodeur aux couches de caractéristiques de son bloc décodeur associé. Un saut de connexion est placé à la fin de chaque bloc de

convolution conduisant à un total de quatre dans notre implémentation.

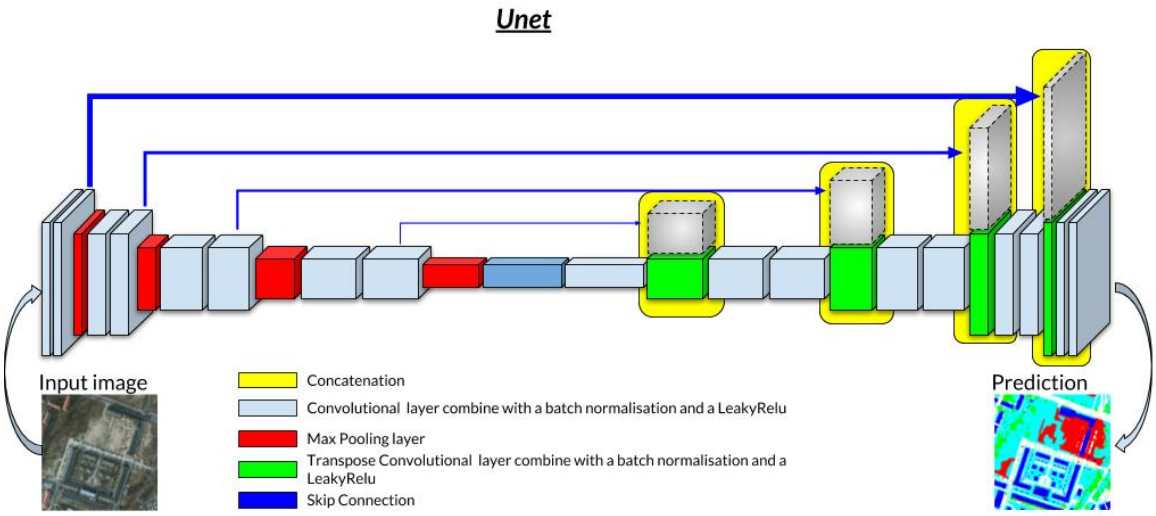


figure 2.5: Le modèle Unet [73].

Tiramisu

Le réseau Tiramisu [40] est une série de blocs denses [34] alternés avec des sauts de connexions et des opérations de transition vers le bas (côté encodage) et de transition vers le haut (côté décodage). L'architecture a un total de 100 couches, ce qui en fait un modèle long à entraîner. Comme le montre la figure 2.6, un bloc dense est composé de 4 couches denses. La première est appliquée à l'entrée pour créer k cartes d'entités, qui sont concaténées à l'entrée. Une deuxième couche dense est appliquée pour créer d'autres couches d'entités, qui sont à nouveau concaténées aux cartes d'entités précédentes. L'opération est répétée quatre fois. La transition vers le haut est une convolution transposée de 3×3 comme *conv-deconv* et *Unet*. Quant à la transition vers le bas, elle utilise un 1×1 conv et un 2×2 max pooling.

Feature Pyramid Net (FPN)

Le *Feature Pyramid Net* (FPN ou PyramidNet) [53] peut être vu comme une version multirésolution du UNet car il y a une hiérarchie pyramidale de prédictions.

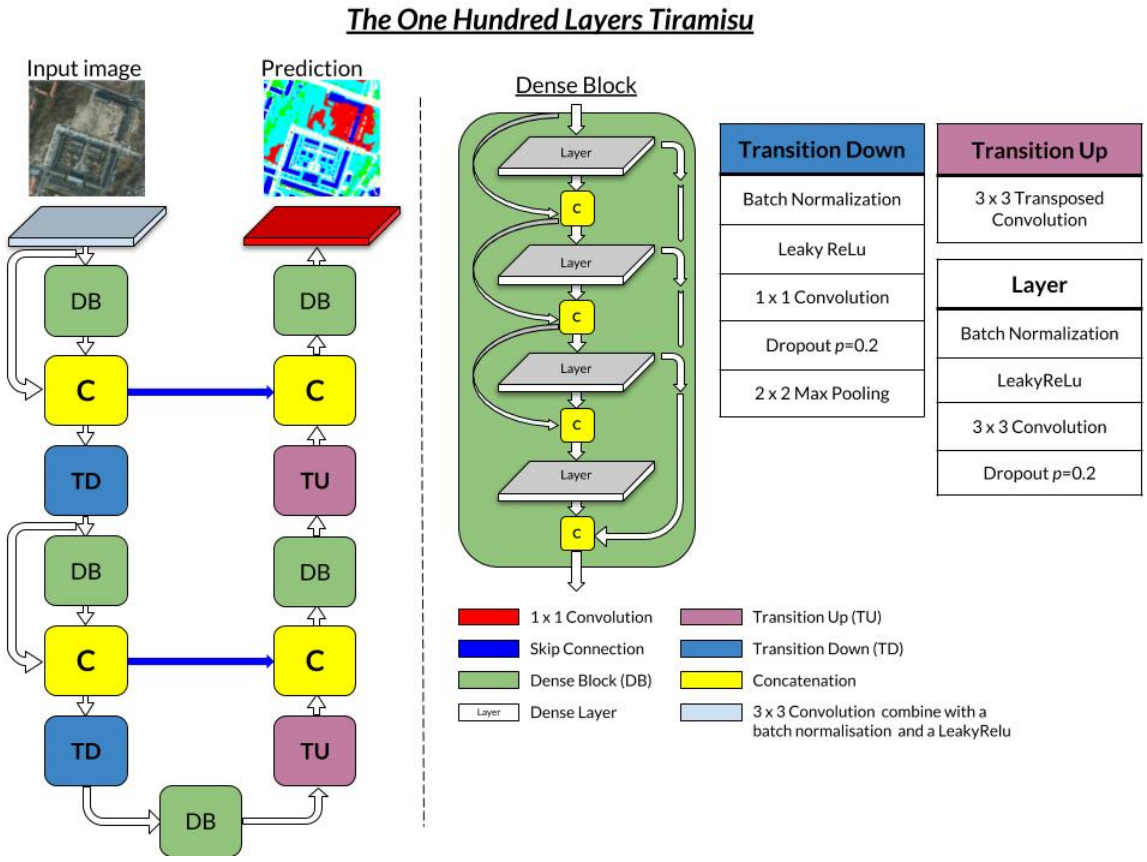


figure 2.6: Le modèle Tiramisu Net [40].

Comme le montre la figure 2.7, cette méthode prend une image à échelle unique en entrée et génère un champ d'étiquettes à la fin de chaque bloc de décodage. L'avantage de cette architecture vient du fait que les champs d'étiquettes prédits par les couches les plus grossières sont plus précis sur les gros objets, tandis que ceux des couches les plus fines sont meilleurs pour segmenter les objets à petite échelle. La segmentation finale se fait en combinant les quatre prédictions par un vote majoritaire.

Bien que les connexions vertes de la figure 2.7 pourraient être des sauts de connexions, *Lin et al.* [53] propose d'utiliser une *connexion latérale*. Une *connexion latérale* est une carte d'activation à résolution plus grossière, dont la résolution spatiale est suréchantillonnée d'un facteur 2 (en utilisant une méthode de suréchantillonnage bilinéaire). La carte suréchantillonnée est ensuite fusionnée avec la carte ascendante

correspondante (qui subit une convolution 1×1 pour réduire ses dimensions de carte d'entités) par addition élément par élément.

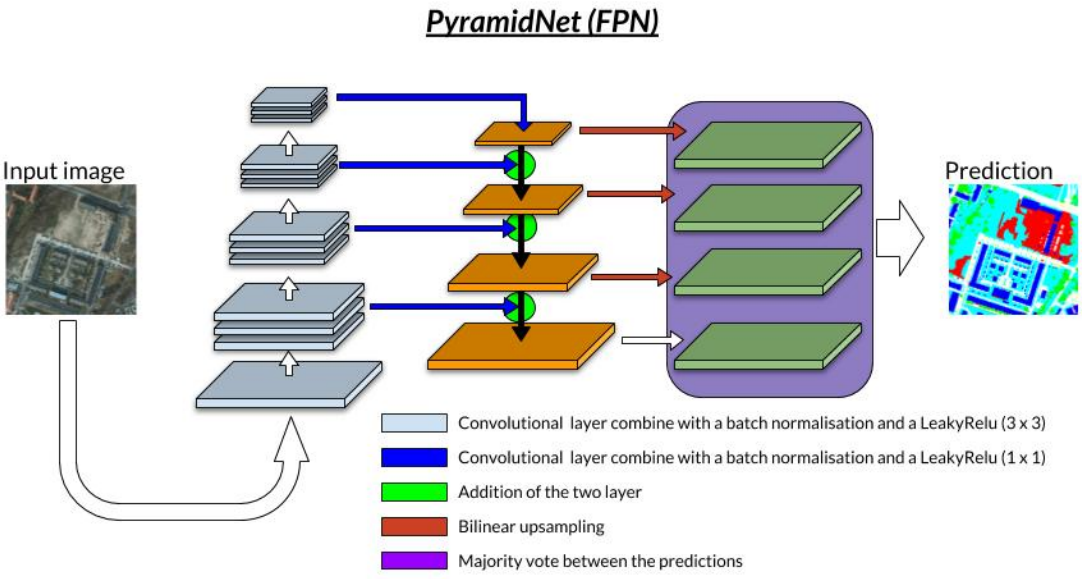


figure 2.7: Le modèle *pyramid network* [53].

DMSMR

Le réseau *dual multi-scale manifold ranking* (DMSMR)[95] est un CNN conçue pour segmenter les images de télédétection. La structure du réseau DMSMR est représentée sur la figure 2.8. Le réseau est principalement composé de cinq blocs convolutifs (le rectangle gris "Couches de convolution dilatées" sur la figure 2.8) et dont la sortie est suréchantillonnée pour produire une carte de segmentation. Dans un esprit similaire au *Pyramid Net*, la sortie de chaque bloc de convolution est envoyée à un ensemble de couches pour produire quatre cartes de segmentation à multiéchelles. Dans l'article original, les auteurs proposent de combiner les cinq résultats avec une méthode dite de classement multiple dont le but est d'imposer un lissage spatial.

Malheureusement, notre implémentation de classement multiple n'a pas réussi (les auteurs n'ont fourni de code pour être reproduits) et nous avons donc décidé de la remplacer par un champ aléatoire conditionnel (CRF) [46] à la fin du réseau. Notez

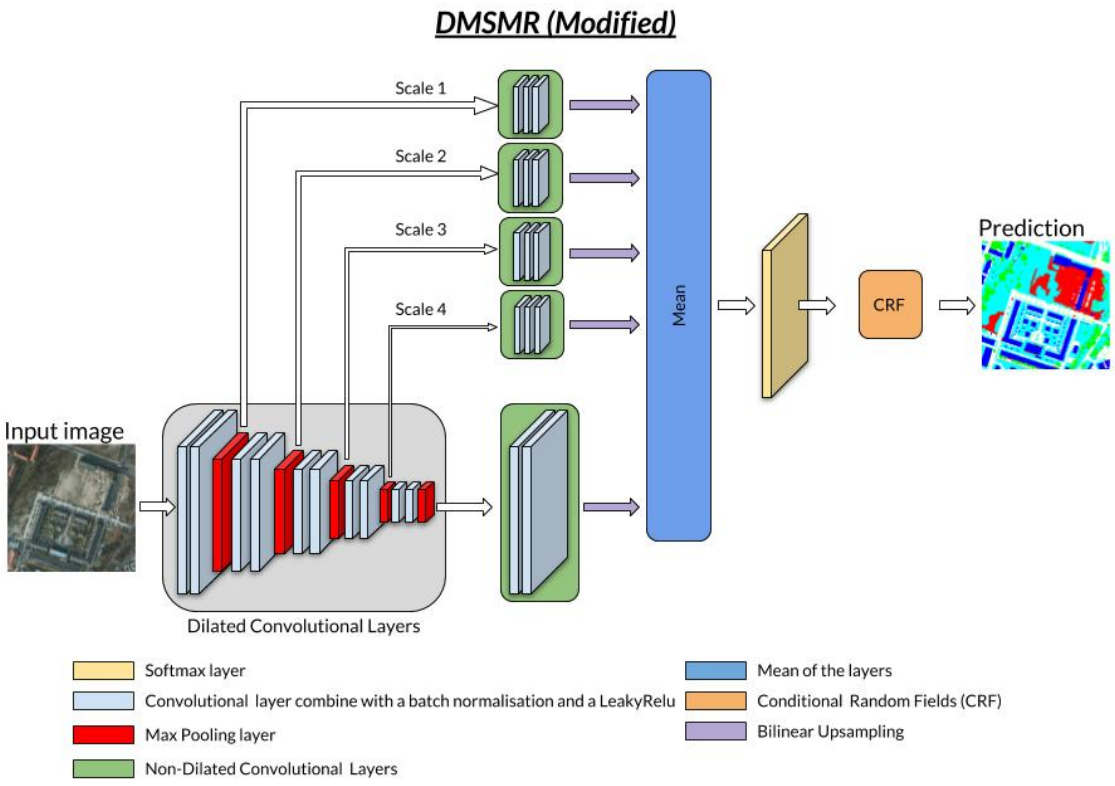


figure 2.8: Le modèle DMSMR [95].

que les CRF sont couramment utilisés pour régulariser spatialement les cartes de segmentation [96, 97, 92] et sont donc un bon substitut au classement d'origine multiple.

ENet

Le réseau *Efficient Neural Network* (ENet)[67] donne la possibilité d'effectuer une segmentation sémantique pixel par pixel en temps réel. Le *ENet* est jusqu'à 18 fois plus rapides que le SegNet[6], nécessite 75 fois moins de FLOP et comporte 79 fois moins de paramètres tout en offrant une précision similaire ou meilleure aux modèles existants. Il a tout d'abord été créer dans l'optique d'être intégré à des véhicules autonomes qui sont l'une des applications où la segmentation sémantique en temps réel est une grande valeur ajoutée. Bien que les précisions continuent d'augmenter, les opérations ne peuvent être déployées en temps réel en raison du grand nombre

d'opérations en virgule flottante et des temps d'exécution identiques. Souvent, les résultats en matière de précision sont comparables et parfois même meilleurs pour *ENet* que pour *Conv-Deconv*, mais avec des temps de calcul grandement réduits.

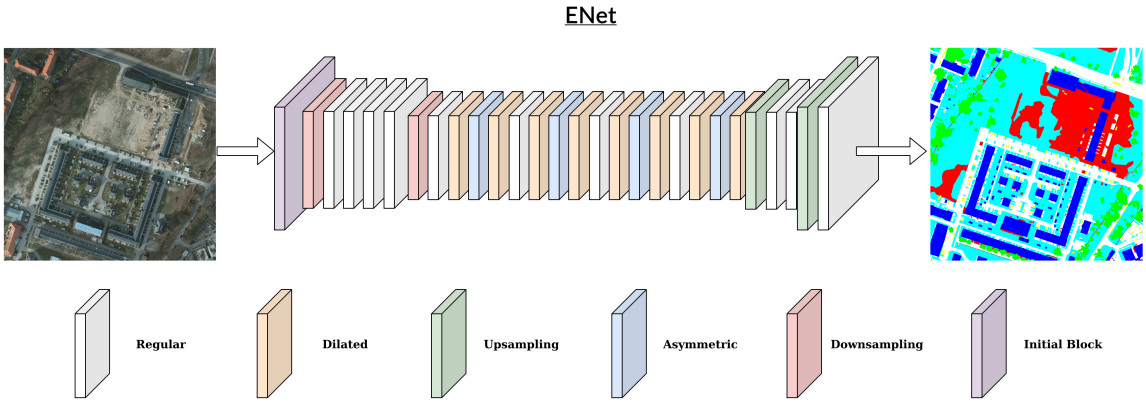


figure 2.9: Le modèle ENet [67].

La figure2.9 montre l'architecture entière du *ENet* qui est largement basée sur celle du ResNet[31]. Cela se résume à une structure avec une base large et plusieurs branches qui se séparent de la base, mais fusionnent également par addition élément par élément. Comme dans l'article original de ResNet[31], ces branches sont appelées *bottlenecks* (voir la figure B.1).

Chapitre 3

Méthodes et résultats complémentaires

Ce chapitre résume le rapport fait pour la compagnie Urthecast [1] dans le cadre d'une subvention d'engagement partenarial obtenue en 2018 [2]. Ce résumé a pour objectif de souligner les forces et les faiblesses des différents points importants dans l'utilisation de l'apprentissage profond sur les images des bases de données **Potsdam** et **Vaihingen**. L'objectif de ce rapport était de répondre aux cinq questions suivantes :

- Quelle architecture de segmentation est la mieux adaptée pour segmenter des images de télédétection ?
- Les modalités d'entrée affectent-elles la précision de la segmentation ?
- Le choix d'une fonction de perte spécifique affecte-t-il la qualité globale des cartes de segmentation produites ?
- Combien d'images annotées sont nécessaires pour entraîner un modèle efficacement ?
- Est-ce que l'entraînement et la prédiction de zones d'images superposées ont une incidence sur les résultats obtenus ?

1. Site internet d'Urthecast : https://www.urthecast.com/
2. Subventions d'engagement partenarial de CRSNG : http://www.nserc-crsng.gc.ca/professors-professeurs/rpp-pp/engage-engagement_fra.asp

- Qu'arrive-t-il lorsqu'un modèle entraîné sur un jeu de données "X" est utilisé sur un autre jeu de données "Y" ?

3.1 Architectures de segmentation pour des images de télédétection

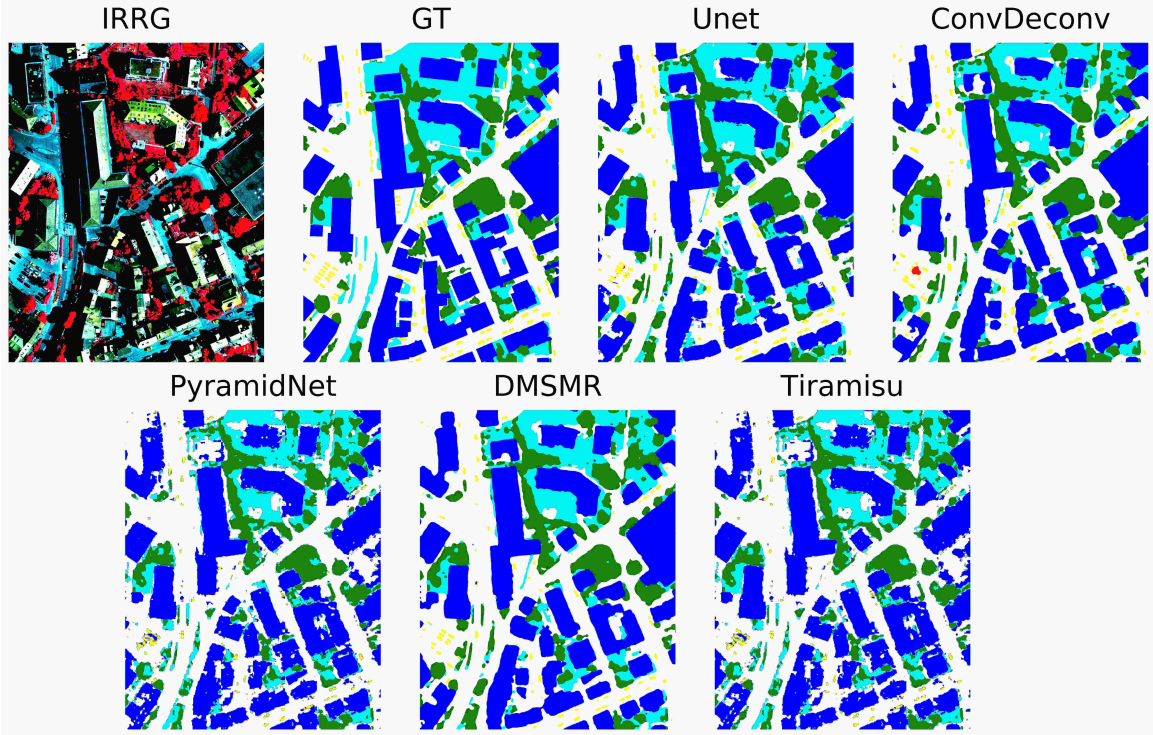


figure 3.1: Exemple de prédictions pour chaque modèle sur une image de l'ensemble de données *Vaihingen* et GT est la vérité terrain.

Ici, nous avons entraîné les cinq modèles décrits précédemment(section 2.4.4) avec une entropie croisée. Dans le monde réel, les ensembles de données d'images satellitaires sont généralement prises avec un ensemble de conditions spécifiques qui ne représentent pas l'intégralité des cas possibles. Cependant, le réseau a pour objectif de performer dans diverses conditions, telles que différentes orientations, emplacements, échelles, luminosité, météo, etc. Pour tenir compte de toutes ces conditions, nous avons modifié les images d'entraînement et de validation avec une augmentation

sur les données. Pour ce faire, nous avons modifié l'orientation des images d'entraînement en leur appliquant au hasard un angle de 0, 90, 180 et 270 degrés.

Les résultats quantitatifs (F1-Score et justesse globale) pour les jeux de données *Postdam* et *Vaihingen* se retrouvent dans les tableaux 3.1 et 3.2. Notez que la dernière colonne des deux tableaux contient les meilleurs résultats de l'état de l'art rapportés sur le site Web de l'ISPRS (c.f **SotA** [3] [4]). Malheureusement, les résultats de l'ISPRS ne contiennent aucune référence aux articles produisant ces résultats. Nous ne connaissons donc pas le contenu exact de ces méthodes "SotA". La figure 3.1 montre la prédiction pour tous les réseaux que l'on a implémentés sur une image du jeu de données *Vaihingen*.

tableau 3.1: Résultats quantitatifs pour le jeu de données **Potsdam**.

	Modèle / Classes	ConvDeconv	Unet	Tiramisu	PytramidNet	DMSMR + CRF	SotA
F1-Score (%)	Résultat global	91.12	91.46	82.96	86.66	91.39	92.0
	Route	93.52	93.59	86.76	89.65	93.65	92.9
	Bâtiment	97.07	97.22	91.11	92.86	97.40	97.3
	Végétation basse	87.17	88.00	81.93	83.52	87.66	86.8
	Arbre	85.46	85.91	77.81	80.20	85.43	87.3
	Voiture	93.44	94.02	76.42	84.50	93.15	95.8
	Obstructions/Background	76.15	76.46	30.65	66.00	76.93	-
Justesse global (%)		91.14	91.49	83.90	86.79	91.41	90.5

Parmi nos méthodes, le Unet[73] a la meilleure justesse globale pour les deux jeux de données, suivi de près par DMSMR[95], alors que leurs *F1-scores* sont très similaires. Les résultats sont également très bons pour chaque classe, excepté pour les classes *Voitures* dans *Vaihingen* et *Background* pour *Potsdam*. Cela peut s'expliquer par le fait que ces deux classes sont sous-représentées dans ces deux ensembles de données (cf. tableau 3.3). Étonnamment, le pyramidnet[53] et le tiramisu[40] n'ont pas bien fonctionné malgré de nombreuses recherches d'hyperparamètres.

3. **Potsdam** http://www2.isprs.org/commissions/comm2/wg4/potsdam-2d-semantic-labeling.html

4. **Vaihingen** http://www2.isprs.org/commissions/comm2/wg4/vaihingen-2d-semantic-labeling-contest.html

tableau 3.2: Résultats quantitatifs pour le jeu de données **Vaihingen**.

	Classes \ Modèle	ConvDeconv	Unet	Tiramisu	PytramidNet	DMSMR+CRF	SotA
F1-Score (%)	Résultat global	87.50	87.93	80.15	81.20	87.76	91.6
	Route	90.07	90.19	84.51	84.27	90.35	93.3
	Bâtiment	92.78	92.68	86.11	84.53	92.84	96.1
	Végétation basse	79.18	80.43	68.74	73.24	79.66	86.4
	Arbre	86.81	87.32	80.24	82.17	87.11	90.8
	Voiture	63.29	66.16	48.69	48.08	61.23	74.6
	Obstructions/Background	83.64	92.70	0.01	66.40	87.83	-
Justesse global (%)		87.57	87.96	80.27	81.28	87.89	?

Nos meilleures méthodes fonctionnent pratiquement aussi bien que le SotA sur le jeu de données *Potsdam*, mais légèrement moins bien sur le jeu de données *Vaihingen*. Malheureusement, puisque ISPRS ne donne aucune référence pour les méthodes utilisées pour générer ces résultats rapportés sur leur site internet, nous ne pouvons pas à coup sûr expliquer ce qui caractérise la méthode SotA. Nous pouvons cependant spéculer que la taille du jeu de données influence les résultats. Le jeu de données *Vaihingen* est beaucoup plus petit que celui de *Potsdam* (694 images contrairement à 8112 images) et que les résultats sont les meilleurs sur chaque classe de *Potsdam*. Nous pensons que nos méthodes auraient pu se rapprocher du SotA si seulement quelques données d'entraînement supplémentaires avaient été fournies avec *Vaihingen*. De plus, il ne serait pas surprenant que la méthode SotA utilise un outil de post-traitement spécifique pour "*remanier*" davantage les résultats (comme imposer une forme polygonale à chaque bâtiment ou d'autres approches du genre), ce qui n'a pas été fait dans notre cas. Pour le reste de ce chapitre, nous utiliserons le Unet[73], car dans notre cas, c'est le résultat qui nous offre les meilleurs résultats.

En outre, tel que mentionné auparavant, la vérité terrain des jeux de données ISPRS est fournie avec et sans contour pour chaque objet. Dans le tableau 3.3, nous rapportons l'impact de l'incorporation ou non des pixels de contours (*On* vs *Off*) lors du calcul des statistiques de prédiction. Comme on peut le voir, ignorer les pixels aux limites de la segmentation (*Off*) donne un "*boost*" non négligeable de 2 à 3% des statistiques de prédiction. Cela n'est pas surprenant, car les CNN sont connus pour être sujets aux erreurs en bordure des objets. Cela souligne également le fait qu'un outil

de post-traitement pour appliquer une forme *a priori* sur l'objet segmenté pourrait améliorer considérablement les résultats. Il serait également possible de prédire les contours dans le réseau et utiliser cette prédiction pour un CRF (post-processing ou directement dans le réseau).

	Contours / Classes	*Potsdam*				*Vaihingen*			
		On	Support	Off	Support	On	Support	Off	Support
F1-Score (%)	Résultat global	89.26	203 513 856	91.46	189 270 677	84.83	17 310 720	87.93	15 666 348
	Route	91.51	58 894 300	93.59	54 616 809	87.27	4 941 465	90.19	4 488 561
	Bâtiment	96.23	54 360 512	97.22	52 714 973	90.63	4 371 376	92.68	4 081 888
	Végétation	85.61	47 951 497	88.00	43 923 472	76.78	3 676 028	80.43	3 262 997
	Arbre	83.11	29 361 477	85.91	27 026 181	84.06	4 084 900	87.32	3 665 015
	Voiture	88.61	3 566 587	94.02	2 672 088	60.23	199 490	66.16	133 681
	Obstructions/Background	72.91	9 379 483	76.46	8 317 154	89.32	37 461	92.70	34 206
Justesse global (%)		89.32		91.49		84.87		87.96	

tableau 3.3: Influence de l'incorporation (On) ou non (Off) des pixels de contours lors du calcul de la justesse pour la méthode Unet[73] avec une perte d'entropie croisée, où *Support* est le nombre de pixels pour chaque classes.

3.2 Relation entre les modalités d'entrée et précision de la segmentation

Dans cette section, le jeu de données **Potsdam** est utilisé, car il contient le plus grand nombre de canaux d'entrée (RGB, IR et DSM). Il est alors plus facile d'effectuer une étude d'ablation pour mesurer l'effet de ces canaux. Les résultats pour Unet[73] sont présentés dans le tableau 3.4.

Comme on peut le constater, bien que les canaux IR-RG ou RGB donnent à peu près les mêmes résultats, la combinaison de tous les canaux en entrée (IR + RGB + DSM) donne de meilleurs résultats. Considérant que le DSM est rarement acquis en imagerie satellitaire, la combinaison des canaux de couleurs et d'infrarouge fournit toujours de meilleurs résultats qu'avec seulement trois canaux de couleur.

	Classes \ Modalité	IRRG	RGB	IRRGB	IRRGB + DSM
F1-Score (%)	Résultat global	87.74	87.37	88.11	**89.26**
	Route	90.11	89.98	90.38	**91.51**
	Bâtiment	94.46	94.56	94.74	**96.23**
	Végétation basse	84.50	83.20	84.74	**85.61**
	Arbre	81.99	81.65	82.21	**83.11**
	Voiture	88.04	88.20	88.66	**88.61**
	Obstructions/Background	68.43	68.14	70.89	**72.91**
Justesse global (%)		87.84	87.41	88.15	**89.32**

tableau 3.4: Résultats pour le Unet[73] sur **Potsdam** avec différentes modalités d'entrée.

3.3 Relation entre les fonctions de perte et la qualité globale des cartes de segmentation produites

Les résultats empiriques pour le modèle Unet[73] sur les deux jeux de données avec la fonction de perte d'entropie croisée[73], la *Dice loss*[20] et la combinaison des deux (*Combo*) sont rapportés au tableau 3.5. Contrairement à ce que nous observons souvent dans d'autres domaines d'application (comme l'imagerie médicale), l'entropie croisée fournit de meilleurs résultats que la *Dice Loss* et la combinaison des deux ne donne aucun avantage apparent (sauf pour les *Routes* sur **Potsdam**). Notons qu'encore une fois de meilleurs résultats sont obtenus avec **Potsdam**, probablement parce que **Potsdam** est un jeu de données plus gros que Vaihingen.

3.4 Nombre d'images annotées nécessaires pour entraîner un modèle efficacement

Le nombre d'images utilisées pour entraîner un modèle est fondamentalement important pour qu'un réseau puisse bien généraliser sur de nouvelles données jamais observées. Il est bien connu qu'un ensemble d'entraînement trop petit induit souvent

Jeux de données		**Potsdam**			**Vaihingen**		
Classes \ Fonction de perte		Entropie croisée	Dice	Combo	Entropie croisée	Dice	Combo
F1-Score (%)	Résultat global	**89.26**	88.86	89.12	**84.83**	84.13	84.28
	Route	91.51	91.12	**91.55**	**87.27**	86.53	87.11
	Bâtiment	**96.23**	95.99	96.09	**90.63**	89.58	90.27
	Végétation basse	**85.61**	85.18	85.55	**76.78**	76.24	75.88
	Arbre	**83.11**	82.97	82.60	**84.06**	83.93	83.28
	Voiture	**88.61**	87.46	88.11	**60.23**	55.25	57.77
	Obstructions/Background	**72.91**	71.06	72.52	**89.32**	82.84	86.08
Justesse global (%)		**89.32**	88.90	89.18	**84.87**	84.19	84.35

tableau 3.5: Résultats pour le Unet[73] entraîné avec trois fonctions de pertes.

des problèmes d'"*overfitting*"(surapprentissage) et des résultats de prédiction de qualité médiocre. C'est probablement pour cette raison que les meilleurs résultats obtenus sont avec **Potsdam** et non avec **Vaihingen**.

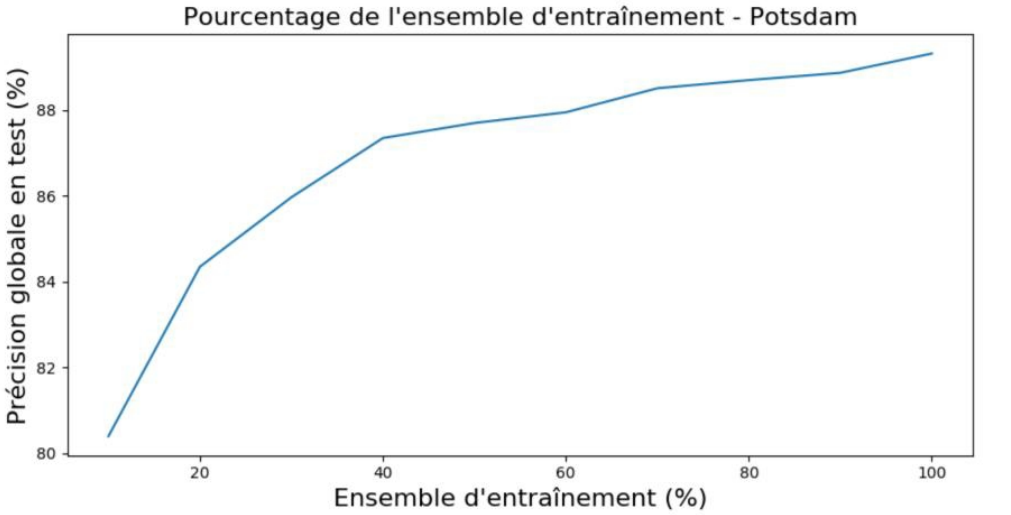
On peut penser que plus un réseau voit de données d'apprentissage, mieux il généralisera à de nouvelles images, mais cela n'est pas entièrement véridique. Il a été démontré à plusieurs reprises qu'un modèle ne pouvait pas dépasser sa propre capacité d'apprentissage. En d'autres termes, même avec un énorme ensemble d'entraînement, la capacité du réseau à généraliser finira par atteindre un certain plateau.

Cette section souligne l'influence du nombre d'images utilisées pour entraîner un modèle sur la précision des tests. Nous souhaitons ainsi mesurer si la taille du jeu de données ISPRS est suffisamment grande et si les performances du modèle choisi ont atteint ou non un plateau. Pour ce faire, **Postdam**, dont le jeu de données d'entraînement contient 8112 images, est utilisé pour entraîner un Unet[73] avec différents sous-ensembles d'images et testé sur un jeu de tests de taille fixe. Les résultats sont représentés au tableau 3.6 et à la figure 3.2.

Sans grande surprise, plus le réseau voit d'images au cours de son entraînement et meilleures sont ses performances au moment des tests. Cependant, comme on peut le

figure 3.2 & tableau 3.6: Précision globale selon le pourcentage du nombre d'images d'entraînement de **Postdam** utilisé pour entraîner un Unet[73].

Training dataset (%)	10	20	30	40	50	60	70	80	90	100
Justesse globale	80,40	84,35	85,97	87,35	87,70	87,95	88,51	88,70	88,87	89,32



voir sur la figure 3.2, la courbe présente une progression régulière sans pour autant atteindre un plateau. Cela indique que la précision globale obtenue lors de l'utilisation de 100% de l'ensemble de données pourrait être améliorée si plus de données annotées avaient été disponibles.

3.5 Entraînement et prédiction de zones d'images superposées

Tel que souligné à la section 3.4, le nombre d'images d'entraînement est primordial pour garantir une bonne précision des tests. Malheureusement, comme **Vaihingen**, bon nombre de bases de données comptent un nombre limité d'images annotées. Dans ces cas, il est possible d'entraîner un modèle avec des "*patchs*" qui se chevauchent tel qu'illustré dans la figure 3.3 afin d'augmenter artificiellement l'ensemble d'entraînement[33]. Dans cette section, un Unet[73] sur divers niveaux de chevauchements d'images de **Vaihingen** et ensuite testés sur **Potsdam**. Cela permet non seulement de mesurer l'efficacité du chevauchement d'images, mais également d'évaluer

dans quelle mesure le transfert d'apprentissage fonctionne avec les images satellites de deux villes différentes. Puisque les images satellites sont de grandes dimensions, nous avons testé les prédictions par "*patch*" et les prédictions d'images complètes (4832x4832) pour comparer les temps de traitement. Les résultats sont rapportés au tableau 3.7.

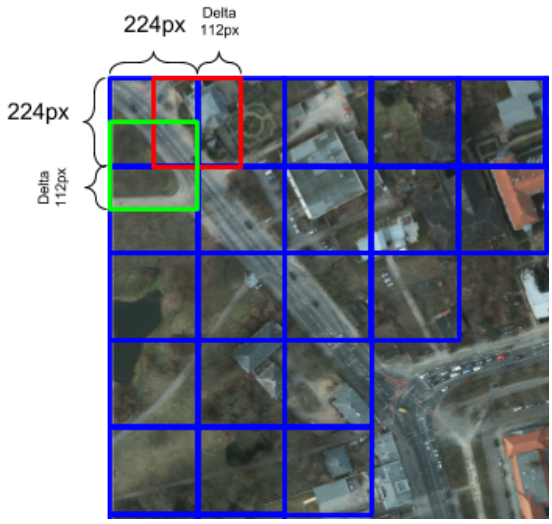


figure 3.3: Exemple de *patch* superposées à 50%.

Comme on peut le constater, l'utilisation d'un chevauchement d'images important permet d'améliorer la précision globale des résultats. Les résultats expérimentaux révèlent qu'un chevauchement de 70% est un point idéal avec une augmentation des résultats de plus de 7% par rapport à l'absence de chevauchement. En outre, la prédiction sur une image de 224x224 pixel ou de 4832x4832 pixels (taille maximale rentrant sur le GPU) n'affecte en aucun cas la précision globale ni les temps de traitement.

Il est surprenant de constater à quel point la précision globale des tests est faible (au mieux 53.72% contre 89.32% au tableau 3.6) considérant que **Potsdam** et **Vaihingen** sont deux villes allemandes visuellement similaires. Ces résultats soulignent à quel point il est difficile de transférer un modèle entraîné d'un jeu de données satellites à

Entraînement		Prédiction			
224x224		224x224		4832x4832	
Chevauchement (%)	nb images	justesse (%)	temps (s)	justesse (%)	temps (s)
0	694	48,87	0,01	48,63	5,58
10	847	45,98	0,01	45,76	5,59
20	1 052	49,74	0,01	50,08	5,57
30	1 344	50,76	0,01	51,06	5,58
40	1 875	48,82	0,01	48,17	5,57
50	2 596	50,07	0,01	49,87	5,60
60	4 015	50,28	0,01	49,63	5,60
70	7 059	55,83	0,01	55,47	5,60
80	15 855	53,72	0,01	52,66	5,62
Nb d'images prédites		16 224		24	

tableau 3.7: Justesse globale sur **Potsdam** pour le modèle Unet[73] entraîner sur **Vaihingen** avec divers pourcentages de recouvrement d'images.

un autre.

3.6 Entraînement d'un modèle sur un jeu de données "X" pour affiner un jeu de données "Y"

Les résultats obtenus dans la section précédente suggèrent qu'il est difficile de transférer un modèle d'un jeu de données à un autre. Nous avons donc voulu mesurer l'impact de l'affinement d'un modèle sur un jeu de données "X" préalablement entraîné sur un jeu de données "Y". Les résultats de l'optimisation d'un réseau par rapport à une initialisation aléatoire sont rapportés au tableau 3.8. Notez qu'aucun chevauchement d'images n'a été utilisé pour cette expérience.

Comme on peut le constater, préentraîner un modèle sur un grand jeu de données tel que **Potsdam**, puis le peaufiner sur un petit jeu de données tel que **Vaihingen** apporte globalement une légère amélioration d'environ 1%. Bien que cela puisse paraître insignifiant, le préentraînement préalable sur **Potsdam** a eu un impact non négligeable sur les petites classes telles que *Voiture* et *Obstructions/Background*, dont

Jeux de données		**Vaihingen**		**Potsdam**	
	Poids / Classes	initialisation aléatoire	pré-entraîné sur **Potsdam**	initialisation aléatoire	pré-entraîné sur **Vaihingen**
F1-Score (%)	Résultat global	84.83	**85.08**	**89.26**	87.73
	Route	87.27	**87.47**	**91.51**	90.21
	Bâtiment	**90.63**	90.52	**96.23**	93.99
	Végétation basse	76.78	**77.09**	**85.61**	84.54
	Arbre	84.06	**84.26**	**83.11**	82.17
	Voiture	60.23	**73.08**	**88.61**	87.87
	Obstructions/Background	73.03	**89.32**	**72.91**	69.55
Justesse global (%)		84.87	**85.09**	**89.32**	87.81

tableau 3.8: L'effet de l'utilisation de poids pré-entraînés par rapport à une initialisation aléatoire.

le score F1 a été augmenté par 13 et 16 points respectivement. Cela souligne clairement que les petites classes d'un petit jeu de données peuvent bénéficier grandement d'un préentraînement sur un jeu de données composé d'un plus grand nombre de pixels par classes. D'autre part, le préentraînement préalable d'un modèle sur un petit jeu de données tel que **Vaihingen** n'a aucun impact positif sur un jeu de données plus volumineux tel que **Postdam** (c.f. les 2 dernières colonnes du tableau 3.8).

Chapitre 4

Utilisation d'images satellitaires partiellement annotées par des modèles d'apprentissage profond

Dans ce chapitre, nous abordons le projet de maîtrise questionnant l'utilisation d'un ensemble de données pauvrement (ou non) annoté avec des particularités différentes pour entraîner un modèle. Nous présentons un concept simple, flexible et général pour l'entraînement de réseau de segmentation d'images satellitaires partiellement annotées. Bien qu'il existe un nombre croissant de bases de données d'images satellitaires annotées, celles-ci ont rarement les mêmes étiquettes de classes. L'objectif premier de l'approche présentée vise à détecter les relations entre différentes classes des différents jeux de données. Pour cela, on divise les étiquettes en deux niveaux hiérarchiques où les différents jeux de données pourront partager les mêmes étiquettes sur un des deux niveaux.

Plus la demande de l'industrie va augmenter, plus les classes d'objets vont augmenter et plus elles seront spécifiques. À chaque ajout d'un nouvel ensemble de données, il est assez long (et coûteux) d'annoter tous les pixels pour toutes les nouvelles classes sur tous les ensembles de données existants. Pour remédier à ce problème, nous proposons une approche qui permet seulement d'étiqueter les classes d'un niveau d'abs-

traction plus élevé regroupant des classes plus spécifiques. Nous expérimentons sur Potsdam et Vaihingen deux ensembles de données de ISPRS, que l'on représente sous forme hiérarchique à la figure 4.1.

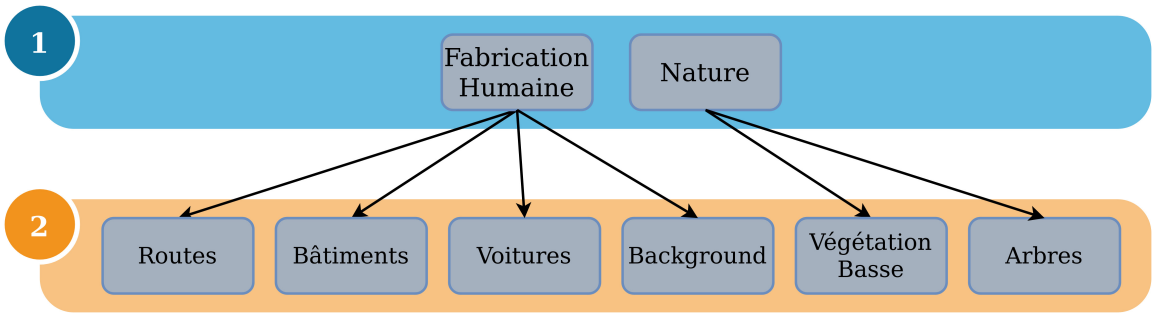


figure 4.1: Illustration de la hiérarchie utilisée pour l'ensemble de données **ISPRS**.

4.1 Introduction

Dans le domaine de la segmentation, il y a une demande croissante de classes d'objets, ainsi qu'une plus grande quantité de données qui s'additionne au fur du temps. En pratique, les nouvelles classes doivent être ajoutées aux images, qui devront être utilisées pour réentraîner le modèle. Cela prend du temps et beaucoup de ressources humaines pour seulement étiqueter les nouvelles images et réétiquetter celles déjà existantes. Cela serait beaucoup plus avantageux si nous pouvions continuellement ajouter de nouvelles données avec une annotation partielle et être capable de générer une annotation complète provenant d'un autre jeu de données.

Dans ce chapitre, nous proposons de résoudre ce problème avec un nouveau concept d'architecture prenant en entrée en mélange de données ayant deux niveaux d'annotation. L'entraînement sur ce jeu de données mixte vise à utiliser deux ou plusieurs ensembles de données étiquetés différemment, mais partageant une certaine logique hiérarchique, pour générer plusieurs niveaux d'annotation, comme représentée à la figure 4.2 avec deux niveaux.

Généralement, un modèle de segmentation est entraîné sur un nombre fixe de classes pour effectuer la segmentation la plus précise possible sur un ensemble de données en particulier. L'approche proposée a pour but de combiner deux bases de données : une base de données partiellement annotée (2 classes) et une base de données avec une pleine annotation (6 classes). L'objectif est donc de généraliser les six classes de l'ensemble pleinement annoté sur l'ensemble partiellement annoté.

figure 4.2: Illustration du concept prenant des données partiellement annotées et complètement annotées pour générer des cartes de segmentation dont les étiquettes de classes reflètent différents niveaux d'abstraction.

La solution la plus simple pour parer à ce problème serait de concaténer toutes les étiquettes des différents ensembles de données. Elle consiste donc à jumeler les étiquettes identiques des différents jeux de données et créer de nouvelles étiquettes pour celles qui se retrouvent dans un seul des ensembles. Dans notre cas, cette approche n'est pas la meilleure, car il est aussi possible que ces différents jeux de données aient été étiquetés par des personnes avec différents points de vue sur le même objet. Par exemple, les buissons du jeu de données de Vaihingen peuvent être étiquetés "arbre", alors que dans Potsdam, ils peuvent être étiquetés "végétation basse". De plus, cette méthode nécessite une annotation complète pour les différents jeux de données et cela va à l'encontre de notre objectif. Considérant ces problématiques, nous proposons une approche composée de nouvelles architectures pour l'entraînement d'ensembles de données croisées spécialement conçus pour la segmentation. Cette dernière se compose

en trois étapes :

1. Créer une relation hiérarchique entre les différentes annotations des ensembles de données (figure 4.1).
2. Générer un jeu de données hybride en conservant toujours les informations hiérarchiques de chaque jeux de données.
3. Entraîner un réseau de segmentation sur l'ensemble de données hybrides dont la structure a été spécialement ajustée.

À titre comparatif, un modèle sans modification sera entraîné à segmenter les six classes originales (routes, bâtiments, voiture, background, végétation basse et arbres) sur l'ensemble Potsdam complètement annoté. Par la suite, nous reprenons les mêmes images de l'ensemble de données Potsdam, mais cette fois avec une annotation partielle. Puisqu'il s'agît de l'ensemble de données avec le plus d'images, l'influence du nombre d'images partiellement annotées sur la justesse des résultats sera plus simple à illustrer. L'ensemble de données d'entraînement sera alors composé d'un mélange des deux ensembles. Ce mélange est composé de l'ensemble Vaihingen complètement annoté et un certain pourcentage prédéfini de l'ensemble Potsdam partiellement annoté. Ce mélange d'ensembles pour l'entraînement est renommé ensemble de données croisées. On termine avec un ensemble de test composé d'images de Potsdam jamais vue par le réseau, sur lequel on génère une prédiction de six étiquettes de classes soit une annotation complète. Pour simplifier les appellations, les ensembles de données sont renommés : **P1**, **P2**, **V1** et **V2**. La première lettre représente le jeux de données (**P** pour Potsdam, **V** pour Vaihingen) et le chiffre représente le niveau hiérarchique tel que vue à la figure 4.1.

P1 : Premier niveau d'identification pour le dataset Potsdam, il réfère à deux classes soient *Fabrication Humaine* et *Nature*.

P2 : Deuxième niveau d'identification pour le dataset Potsdam, il réfère aux six classes originales soient *Bâtiments*, *Routes*, *Voitures*, *Arbres*, *Végétation Basses* et *Backgroud*.

V1 : Premier niveau d'identification pour le dataset Vaihingen, il réfère à deux classes soient *Fabrication Humaine* et *Nature*.

V2 : Deuxième niveau d'identification pour le dataset Vaihingen, il réfère aux six classes originales soient *Bâtiments*, *Routes*, *Voitures*, *Arbres*, *Végétation Basses* et *Backgroud*

4.2 Entraînement avec plusieurs jeux de données

Pour chaque ensemble de données, un ensemble d'étiquettes est fourni et dans la majorité des cas, ces étiquettes sont uniques à l'ensemble de données. Toutefois, on peut regrouper chacune des classes selon des ensembles logiques, car chaque élément fait partie d'un plus gros ensemble : un chat fait partie de la famille des mammifères qui à son tour fait partie de la catégorie animaux. Il est donc hautement probable que les groupes de classes d'un jeu de données *"A"* contiennent les mêmes groupes de classes d'un autre ensemble de données *"Z"*. Si ces classes sont regroupées dans les mêmes groupes on se retrouve avec notre système à niveaux hiérarchiques.

Lors de la mise en place du système hiérarchique pour l'ensemble de données ISPRS (section 1.2.3), nous avons constaté certaines problématiques :

1. **Modalités d'entrée différentes** : dans le cas de Potsdam, les orthophotos sont constituées de **RGBIR** (*rouge - vert - bleu - infrarouge*) contrairement à Vaihingen qui ne contient que trois canaux (*infrarouge - rouge - vert*).
2. **Nombre d'images différents** : Potsdam est composé de 24 images annotées avec 6000×6000 pixels chacune, alors que Vaihingen n'a que 16 images avec dimensions variables. Cela crée un déséquilibre au niveau des proportions des ensembles dans l'ensemble de données croisées.
3. **Résolution différente** : les deux ensembles de données ont des images de résolutions différentes au sol : Potsdam a une résolution au sol de 5cm tandis qu'elle est de 9cm pour Vaihingen.

Pour la première problématique, il existe une solution simple pouvant être apportée. Sélectionner seulement les canaux *infrarouge - rouge - vert* de l'ensemble Potsdam lors de la lecture des images règle le problème de modalités d'entrée différentes.

Pour ce qui est de la problématique du nombre différent d'images, le déséquilibre créé jouera en notre faveur pour cette étude. Puisque l'on prend l'ensemble Vaihingen comme ensemble de bases pleinement annoté, cela laisse l'ensemble de Potsdam, comportant le plus d'images, comme ensemble partiellement annoté qui sera ajouté à l'ensemble de données croisées. Ainsi, il y aura plus d'images entre les différents pourcentages de l'ensemble de données ajoutées à l'ensemble de données croisées.

La problématique de différentes résolutions pourrait être solutionnée par l'utilisation de modèle de super-résolution ([48]), mais cela sera pour une autre étude. De plus, cela nous permettra d'observer le comportement du réseau lorsqu'il est confronté à des images similaires, mais de résolutions différentes.

4.3 Expérimentations

4.3.1 Modèles

Notre solution pour combiner des données partiellement et complètement annotées implique deux nouvelles architectures de segmentation. Pour tester nos hypothèses, nous implémentons aussi une architecture qui servira de *baseline*. On notera **A1** l'architecture sans modification, **A2** l'architecture avec les modifications les plus simples et **A4** l'architecture la plus complexe.

Le modèle de segmentation choisi est le Enet[67]. Toutefois, nous avons aussi implémenté le Unet[73] à des fins de comparaisons. De plus, le Unet[73] comprend des sauts de connexions reliant l'encodeur au décodeur, et il sera intéressant d'observer si cela joue un rôle important dans la différenciation des niveaux hiérarchiques.

Architecture A1 : Comme mentionné plus haut, il s'agit du Enet, qui servira de standard afin d'évaluer les performances des deux autres architectures. On illustre l'architecture à la figure 4.3, où à gauche est représentée la formulation probabiliste et un modèle graphique de l'approche à droite. Dans ce cas, il est seulement question de $P(x)P(y_b|x)$, on passe d'une image x à la segmentation de toutes les étiquettes

de classes y_b. De façon générale, les modèles de segmentation sont représentés par un encodeur (bleu) et un décodeur (vert), suivis par une couche entièrement connectée (fc) et d'un *softmax* pour la prédiction.

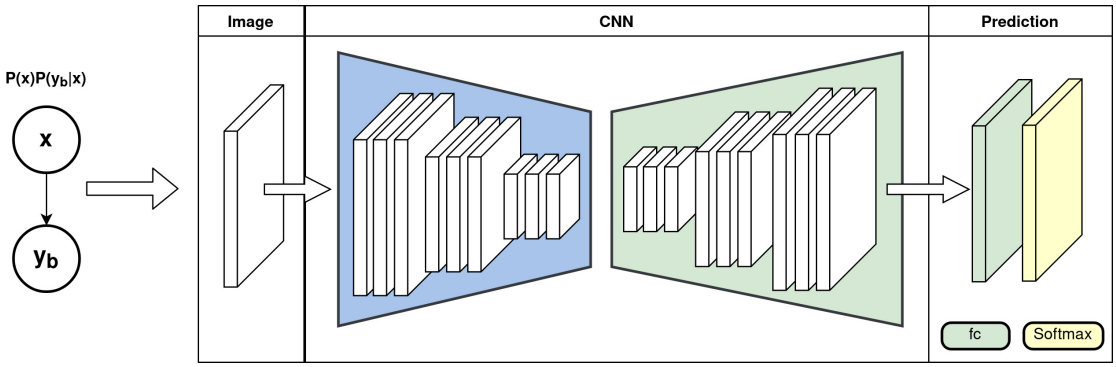


figure 4.3: Illustration du concept de l'architecture $A1$

Architecture A2 : Cette architecture est la façon la plus simple pour gérer plusieurs niveaux d'annotations. Au lieu de prédire un seul niveau d'annotations comme **A1**, l'architecture **A2**(figure 4.4) prédit indépendamment chaque niveau. Mathématiquement, on rajoute l'élément probabiliste $P(y_a|x)$ qui est la segmentation des étiquettes de classes partielles (y_a), on se retrouve alors avec $P(x)P(y_a|x)P(y_b|x)$. De façon visuelle, on duplique les deux couches de la prédiction, soit la couche entièrement connectée (fc) et le *softmax*. L'information contenue dans la dernière couche du

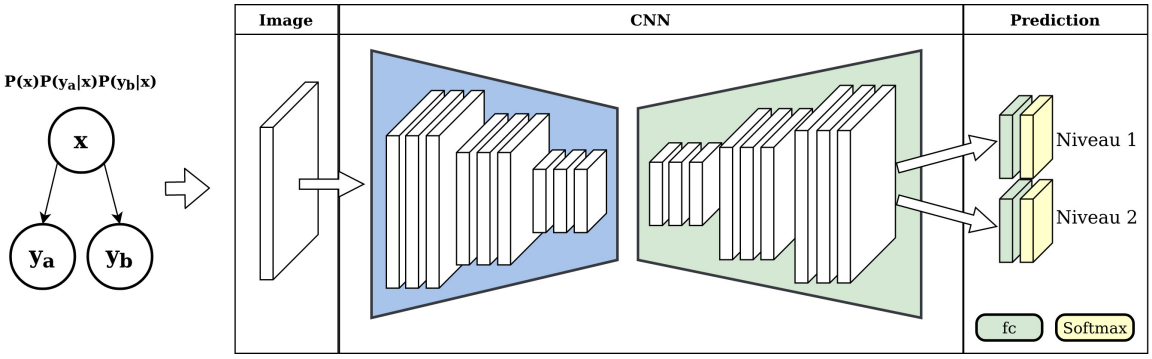


figure 4.4: Illustration du concept de l'architecture $A2$

décodeur sera alors envoyée à chacun de ces blocs. La couche entièrement connectée

de chacun des blocs est implémentée avec couche de convolution dont la dimension de sortie est le nombre d'étiquettes de classes de chaque niveau hiérarchique.

Architecture A4 : Pour cette architecture, on a décidé de mettre toutes les chances de notre côté pour aider le réseau à différentier les deux niveaux hiérarchiques. Comme pour **A2**, l'architecture **A4** illustrée à la figure 4.5 prend en considération les deux niveaux d'annotations. Par contre, au lieu de seulement dupliquer le bloc de prédiction, on duplique aussi le décodeur. L'encodeur fera alors parvenir l'information à deux décodeurs, chacun ayant leur propre bloc de prédiction. Pour s'assurer que le niveau 2 (annotation complète) ait l'information du niveau 1 (annotation partielle), on relie les prédictions entre elles, passant l'information de la couche entièrement connectée (fc) du niveau 1 à celle du niveau 2. Cela change aussi la formulation probabiliste en changeant l'élément $P(y_b|x)$ pour $P(y_b|x, y_a)$, car cette fois la segmentation complète dépend de l'image d'entrée et de la segmentation partielle. La formulation probabiliste de **A4** revient alors à $P(x)P(y_a|x)P(y_b|x, y_a)$.

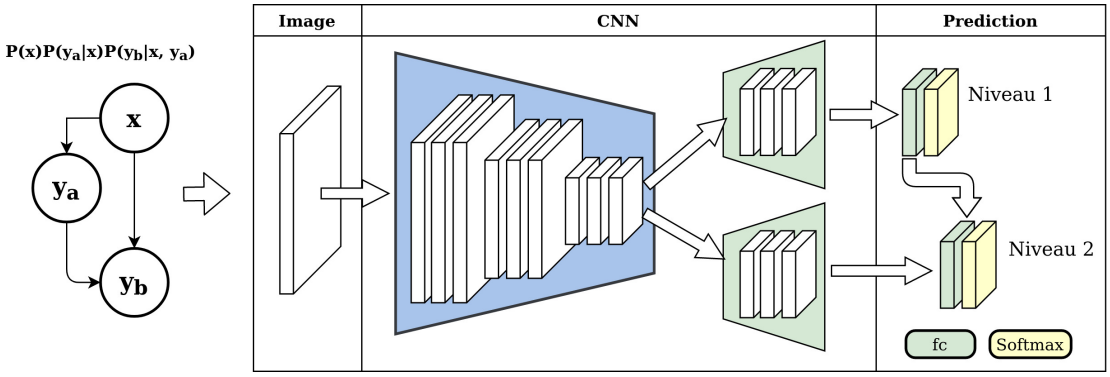


figure 4.5: Illustration du concept de l'architecture $A4$

Les figures B.2, B.3 et B.4, retrouvées en annexe, représentent le Enet[67] utilisé avec les architectures **A1**, **A2** et **A4**. La figure B.1 explique la composition des différentes couches utilisées pour le Enet[67].

4.3.2 Nos expériences

Rappelons que notre objectif principal est le transfert de connaissances. Néanmoins, nous avons l'occasion d'explorer plusieurs autres aspects en lien avec cet objectif, dont l'influence du nombre de données annotées nécessaires au bon fonctionnement du réseau.

> Note, il est important de comprendre que lorsqu'on utilise les termes **P2** et **V2**, cela veut dire par défaut que l'image comprend aussi l'annotation **P1** pour **P2** et **V1** pour **V2** (voir figure 4.6). Car si on a en main une annotation complète, on connaît son annotation partielle. Par exemple, si on a des "arbres" (niveau 2), on sait que celle-ci fait partie du groupe "nature" (niveau 1).

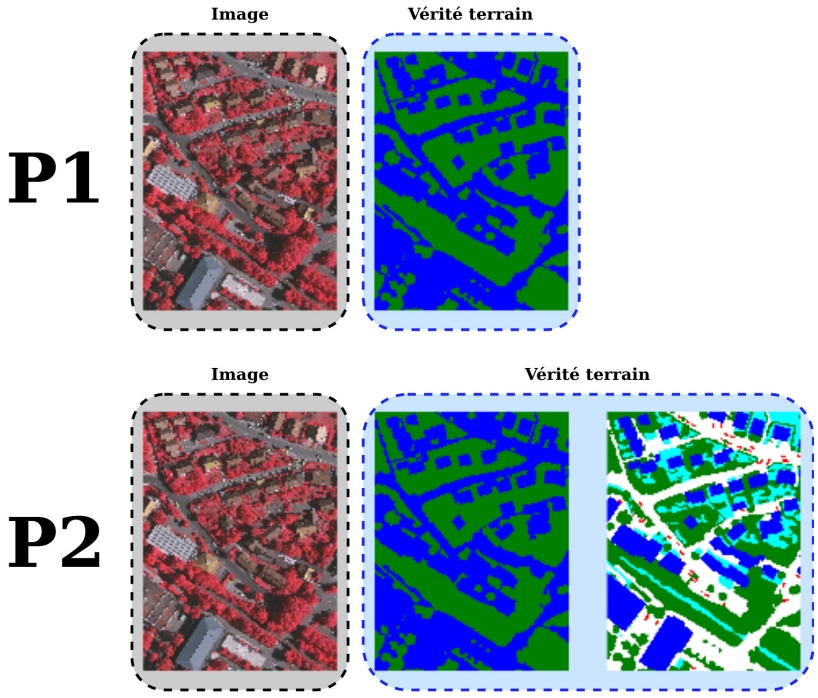


figure 4.6: Illustration de la composition des ensembles **P1** et **P2**. Cette composition s'applique aussi pour **V2**.

La première étape de nos expériences sera l'introduction du jeu de données **P1** au jeu de données **V2**, mais avec un certain pourcentage du jeu de données **P2**. Cela nous permettra d'observer la pertinence d'une annotation complète sur un jeu de données ayant déjà une annotation partielle. La gamme de pourcentages utilisés sera {0%, 6%, 12%, 35%, 53%, 71%, 94%, 100%}, correspondant à {0, 1, 2, 6, 9, 12, 16, 17} images. Alors à 0% le jeu de données d'entraînement ne contient que **P1** et **V2**, et à 100%, il est constitué de **P2** et **V2**. Le tout sera testé sur un ensemble issu de Potsdam.

La deuxième étape porte sur le transfert de connaissances, car on utilisera seulement des images de l'ensemble **P1**. Cette fois, ce que l'on cherche à observer, c'est comment l'ensemble Potsdam est capable d'apprendre l'annotation complète de Vaihingen, alors que Potsdam n'est annoté que par deux étiquettes de classe. De plus, en réutilisant la même gamme de pourcentages qu'au paragraphe précédent, on sera en mesure d'observer l'influence du nombre d'images sur la justesse du modèle, comme dans le chapitre 3, mais cette fois les images n'ont pas une pleine annotation. Alors, pour cette étape, 0% veut dire que le jeu de données d'entraînement ne contient que **V2**, et à 100%, il est constitué de l'intégralité de **P1** et **V2**. Encore une fois, le résultat de l'entraînement sera testé sur Potsdam.

Pour chacune des expériences réalisées dans le cadre de cette maîtrise, nous avons décidé de répéter chaque cycle d'entraînement trois fois. Dans le chapitre 2, on avait vu que l'apprentissage profond est basé sur les statistiques, cela veut dire que le résultat final n'est pas certain et qu'une petite variation dans les données d'entrées peut donner un résultat complètement différent. Sans compter que si le modèle est tombé dans un minimum local lors d'un entraînement, il performera moins bien que si on relance l'expérience et qu'il converge vers le minimum global. Lors d'un entraînement, il y a beaucoup de facteurs qui ne seront pas les mêmes si l'on répète l'expérience une seconde fois. Alors pour être un peu plus convaincu que les résultats obtenus ne sont pas issus du hasard, on reproduira nos expériences trois fois et on moyennera les trois résultats en calculant l'écart type au passage. Cette approche va nous permettre de voir si nos expériences sont stables ou si certaines situations font que le réseau a de

la difficulté à généraliser.

4.3.3 Configuration et entraînement

Tout le code pour ces expériences est écrit en Python et on utilise la bibliothèque d'apprentissage profond PyTorch pour construire les modèles Enet[67] et Unet[73]. On divise les images des deux ensembles de données en patches de 256×256 pour ne pas dépasser la mémoire disponible dans les cartes graphiques.

Toutes les expériences sont effectuées sur 50 époch. Pour toutes ces expériences, nous utilisons l'optimiseur SGD avec un *momentum* de 0.9, un *weight decay* de 5×10^{-4}, un taux d'apprentissage initial de 0.01 et une décroissance du taux d'apprentissage aux époch {25, 35, 45} d'un facteur 0.1. Nous utilisons aussi une taille de lot (*batch size*) de 32 pour ces expériences. Lors des entraînements, nous utilisons la fonction de perte d'entropie croisée 2D. Par contre, pour aider le réseau à mieux apprendre sur les différents niveaux hiérarchiques, on change les poids des fonctions de pertes lors de l'entraînement. Pour l'architecture **A1**, aucun changement n'est apporté, car cette dernière a seulement un niveau hiérarchique.

Puisque **A2** et **A4** ont deux sorties chacune, on utilise deux fonctions de pertes, soit une pour chaque sortie. Cela revient à avoir une fonction de perte pour chaque niveau hiérarchique, normalement on additionnerait les deux fonctions de perte et le réseau apprendrait les deux niveaux en même temps. Après quelques tests, nous avions remarqué que le réseau apprend au premier niveau hiérarchique rapidement, alors on est venu avec une méthode qui attribue des poids aux deux fonctions de perte, cela permet au réseau de balancer son apprentissage. La méthode mise au point ce résume à attribuer un certain pourcentage à chaque fonction, un plus haut pourcentage sur le premier niveau pour commencer et un plus faible pour le deuxième niveau. On réduit progressivement le pourcentage du premier niveau pour atteindre 0%, ce qui revient à ce concentré seulement sur le deuxième niveau hiérarchique, car on garde une somme des deux pourcentages égale à 100% en tout temps, alors en réduisant le premier niveau on augmente l'importance du deuxième. On retrouve la répartition des poids selon les époch dans le tableau 4.1.

Époch	Poids pour le niveau 1	Poids pour le niveau 2
0 à 10	98%	2%
11 à 16	60%	40%
17 à 20	20%	80%
21 à 50	0%	100%

tableau 4.1: Poids des différents niveaux hiérarchique pour les fonctions de pertes lors de l'entraînement.

4.3.4 Résultats

Entraînement avec annotation P2 (Enet)

À la section précédente, nous avons mentionné que nos expériences se divisent en deux étapes. Dans cette section, nous présentons les résultats obtenus lors de la première étape. Cependant, cette dernière sera seulement effectuée avec le modèle Enet[67] (et les deux architectures modifiées) par manque de temps.

Petit rappel pour le déroulement de cette expérience :

- On entraîne **A1** avec une annotation complète de Vaihingen pour ensuite prédire une annotation complète sur l'ensemble de tests de Potsdam (0% de **P2**). On répète l'expérience en ajoutant un certain pourcentage de l'ensemble d'entraînement Potsdam avec une annotation complète, le pourcentage est préétabli par une gamme prédéfinie auparavant. On ne tient pas compte des images annotées partiellement **P1**, car **A1** n'est pas conçu pour pouvoir les traiter.
- On entraîne **A2** [1] avec une annotation complète et partielle de toutes les images de Vaihingen et toutes les images de Potsdam annotées partiellement **P1** (0% de **P2**). Ensuite, on prédit une annotation partielle et complète sur l'ensemble de test de Potsdam. Comme pour **A1**, on répète l'expérience avec un certain pourcentage de **P2**.
- On entraîne **A4** [2], comme pour **A2**, avec une annotation complète et partielle de toutes les images de Vaihingen et toutes les images de Potsdam annotées partiellement **P1** (0% de **P2**). Ensuite, on prédit une annotation partielle et

1. Voir la figure B.3 pour son implémentation sur le Enet.
2. Voir la figure B.4 pour son implémentation sur le Enet.

complète sur l'ensemble de test de Potsdam. Comme pour **A1** et **A2**, on répète l'expérience avec un certain pourcentage de **P2**.

On répète ces expériences trois fois pour obtenir une moyenne et un écart type du *F1-score* pour chaque test (tableau 4.2).

Pourcentage du jeux de données P2 (%)	**A1**	**A2**	**A4**
0%	32.83 ± 6.19%	57.79±1.16%	55.00± 6.28%
6%	63.48 ± 1.89%	71.08±1.47%	74.04± 1.49%
12%	78.46 ± 0.21%	80.73±0.28%	81.31± 0.61%
35%	84.76 ± 0.34%	84.33±0.29%	85.45± 0.10%
53%	85.62 ± 0.08%	85.72±0.38%	85.73± 0.34%
71%	87.20 ± 0.12%	87.09±0.08%	87.55± 0.16%
94%	87.67 ± 0.26%	87.65±0.19%	87.79± 0.30%
100%	87.79 ± 0.09%	87.85±0.17%	87.46± 0.18%

tableau 4.2: *F1-score* des résultat de test sur l'ensemble **P2** et leurs incertitudes pour le Enet[67].

Le tableau 4.2 présente les résultats moyens des entraînements de nos trois réseaux et ces derniers sont illustrés à la figure 4.7. Le premier constat que ce tableau nous permet de faire se trouve sur la première ligne, à 0% de **P2**. Cette ligne nous dit que pour la colonne **A1**, le résultat est celui où le réseau n'a vu que des images de Vaihingen et a essayé de prédire sur un ensemble de données Potsdam qu'il n'avait jamais vu. Lorsqu'on regarde les deux autres colonnes, **A2** et **A4**, les résultats ont augmentés jusqu'à 24.96% et 22.17% respectivement, cette augmentation est due à l'ajout de l'ensemble de données partiellement annoté de Potsdam. Alors, le premier constat intéressant est que l'ajout d'un ensemble de données partiellement annoté augmente de façon significative les performances en généralisation. De plus, si l'on prend l'écart type de **A2** à 0%, on observe qu'il est beaucoup plus petit que celui de **A1** et **A4**. Pour la ligne du 0% de **P2**, l'architecture **A2** s'est avérée la plus efficace. Par contre, à partir de 35%, aucune différence n'est notable.

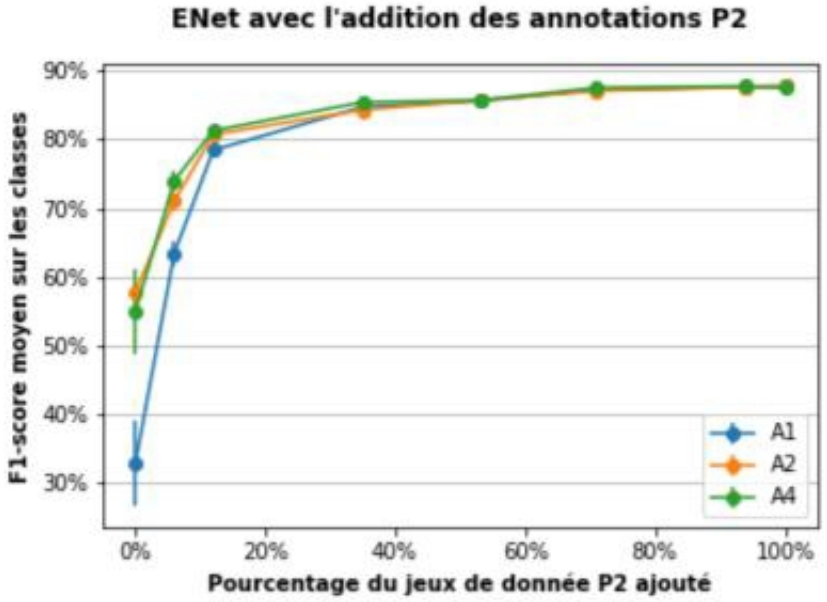


figure 4.7: *F1-score* des résultat de test sur l'ensemble **P2** du modèle Enet[67] sur les trois architectures.

Notre deuxième constat s'observe mieux sur la figure 4.7, car il s'agit de la vitesse à laquelle l'architecture standard **A1** rattrape les résultats obtenus avec **A2** et **A4**. On avait vu au premier constat que le fait d'ajouter seulement **P1** donnait un net avantage aux réseaux, mais dès que l'on ajoute 6% de l'ensemble Potsdam complètement annoté, **A1** double pratiquement ces performances, passant de 32.83% à 63.48%. Toutefois, à 6% **A1** est toujours derrière **A2** et **A4** de près de 10%. Par contre, lorsque **A1** est entraîné avec 12%, l'écart avec **A2** et **A4** est de seulement 3%. À 35%, l'architecture standard avec aucune notion d'hiérarchie de classes rattrape les deux autres architectures, qui elles voient en plus une annotation partielle sur le reste des données de Potsdam. Pour la suite des pourcentages de **P2**, les trois architectures ont des résultats similaires.

Le dernier constat de cette partie se fait sur les résultats de nos deux architectures. Comme mentionné au dernier constat, à partir de 35% d'ajout de **P2**, les trois architectures performent de la même façon, pourtant **A2** et **A4** continue de voir le reste de l'ensemble Potsdam avec une annotation partielle. Techniquement, celles-ci voient

plus d'images du nouvel ensemble, alors pourquoi performent-elles de la même façon ? On ne sait pas vraiment, peut-être que l'architecture **A1** a appris assez sur l'ensemble Potsdam pour bien généraliser sur celui-ci. Il se peut que **A2** et **A4** aient atteintes leur limite d'apprentissage sur le facteur hiérarchique. De plus, les résultats obtenus par **A2** et **A4** ne sont pas très différents les uns des autres, à l'exception des points à 0% et 6%. **A4** performe mieux à 6% de **P2**, même si ceux-ci sont relativement proches au vu de leurs moyennes et écarts-types. Par conséquent, il semble que **A4** a un petit avantage comparer à **A2**, mais seulement pour 6% après cela, il ne semble plus avoir de grandes différences entre les deux architectures. Alors utiliser une architecture plus complexe comme **A4** semble légèrement plus performante qu'une architecture plus simple comme **A2**. Cependant en termes de temps d'entraînement il y a un gain considérable à utiliser une architecture simple comme **A2**.

On en a très peu parlé durant l'évaluation des résultats, mais les écarts types semblent se stabiliser relativement rapidement. Après seulement 12%, tous les écarts types sont descendus sous la barre du 1%, laissant apparaître que même avec peu de données pleinement annotées les prédictions ne vont pas beaucoup différer d'un entraînement à l'autre. Contrairement à 0% de **P2**, où l'écart type était d'environ 6% pour les architectures **A1** et **A4**. Le manque de données annotées complètement rendait les réseaux plus instables lors de la prédiction sur un nouvel ensemble.

Un des bons côtés d'utiliser un deuxième ensemble de données est de pouvoir ajouter plus d'éléments dans des étiquettes de classes souvent moins bien représentées. Dans le cas des réseaux de segmentation, on parle du nombre de pixels représentant une classe. Dans notre cas, on sait que Vaihingen représente une ville plus rurale que Potsdam. Il est donc normal que certaines classes comme *voitures* et *obstructions/background* soient moins bien représentées dans l'ensemble de données contrairement aux *arbres*. C'est ce que le tableau 4.3 nous montre, avec le pourcentage des pixels appartenant à chacune des classes pour les ensembles d'entraînement de Vaihingen et Potsdam.

On peut voir avec le tableau 4.3 que la classe *obstructions/background* est celle qui aurait le plus à bénéficier de l'ajout de l'ensemble de données **Potsdam**. Justement,

Classes	Bases de données **Vaihingen (V2)**	**Potsdam (P2)**
Routes	25.56%	24.58%
Bâtiments	23.41%	22.91%
Végétation basse	19.63%	25.38%
Arbres	20.27%	15.03%
Voitures	0.84%	1.09%
Obstructions/ Background	0.84%	3.98%
Indéfinie	9.46%	7.04%
Nombre de pixels	58 538 632	612 000 000

tableau 4.3: Pourcentage de pixels annotés par classe pour les ensembles d'entraînement.

si l'on regarde l'effet de l'ajout de l'annotation **P2** sur le *F1-score* de cette classe, on se retrouve avec la figure 4.8. L'ajout d'un nombre plus important de pixels annotés **P2** ne semble pas améliorer les résultats de façon plus substantielle que ce que l'on observe pour la moyenne des classes (figure 4.7). Autre que le comportement de **A2** lorsque l'on ajoute 6% de **P2** que nous ne comprenons pas bien, et de l'écart type très important pour **A4** à 0%, l'allure générale des courbes reste très similaire à celles de la figure 4.7.

Alors, est-ce que l'on obtient de meilleurs résultats sur les classes avec peu de pixels annotés en ajoutant plus de pixels d'un nouvel ensemble de données ? Oui. Mais est-ce qu'une architecture comportant une implémentation hiérarchique aide ? Pas vraiment plus que les classes avec un nombre de pixels annotés déjà important.

Transfert de connaissances avec annotation P1 seulement (Enet)

Dans cette partie, on présente les résultats obtenus lors de la deuxième étape de nos expériences. On explore l'aspect du transfert de connaissances avec l'architecture **A2** du modèle Enet[67]. On a pris trois points dans la gamme de pourcentages pour **A4**, et avons comparé les résultats avec **A2**, mais sans réelles différences. Alors, pour des questions de temps, nous avons fait le choix de ne pas effectuer l'expérience au

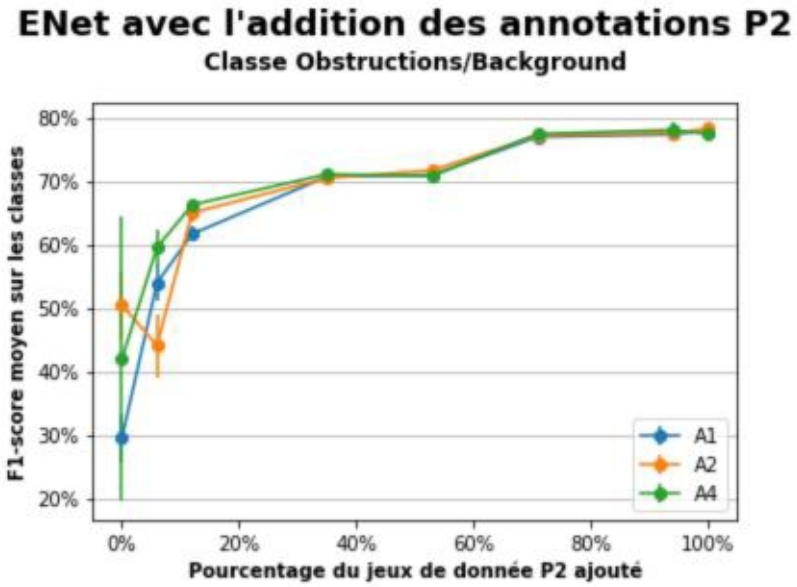


figure 4.8: Résultat du *F1-score* de la classe *Obstructions/Background* du modèle Enet[67] sur les trois architectures.

complet avec **A4**. Nous allons mettre **A1** de côté aussi, car son architecture n'est pas conçue pour être entraînée avec des images annotées partiellement.

Le déroulement de cette expérience est simple, on entraîne **A2** avec une annotation complète de Vaihingen pour ensuite prédire une annotation complète sur l'ensemble de test de Potsdam (0% de **P1** et **P2**). On répète l'expérience en ajoutant un certain pourcentage de l'ensemble d'entraînement Potsdam avec une annotation partielle (**P1**), utilisant la même gamme de pourcentages que lors de la première expérience. Comme pour la partie précédente, on répète l'opération trois fois pour obtenir une moyenne et un écart type du *F1-score* pour chaque entraînement (figure 4.9).

Pour le transfert de connaissances, nous voulions principalement regarder l'influence de l'ajout de données partiellement annotées sur le *F1-score* des prédictions d'une annotation complète. On rappelle que les images **P1** ne sont annotées qu'avec deux classes simples : fabrication humaine et nature.

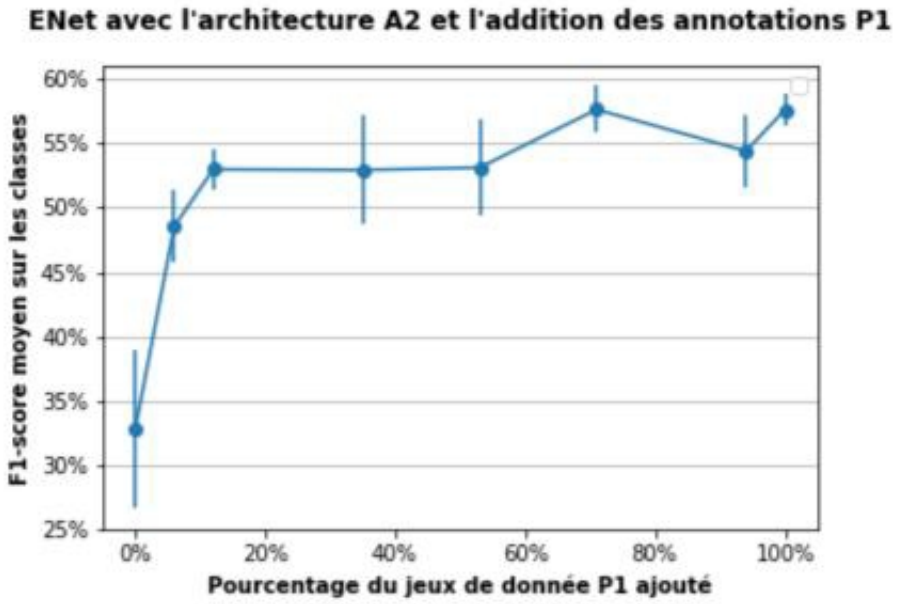


figure 4.9: Résultat moyen du *F1-score* de la moyenne des trois entraînements du modèle Enet[67] sur l'architectures $A2$.

Le premier constat que l'on peut faire de la figure 4.9 est sa similarité avec la courbe avec la figure 4.7. Ce qui nous mène à se dire que l'ajout d'images annotées (partielle ou complète) d'un nouvel ensemble de données aide grandement à la prédiction sur des images issues du même ensemble. Toutefois, après un certain pourcentage du nouvel ensemble les performances du réseau commencent à se stabiliser. Il n'est donc pas nécessaire d'annoter un nouvel ensemble de données complètement pour obtenir des résultats acceptables sur celui-ci, sauvant ainsi beaucoup de ressources lors de l'incorporation d'un nouvel ensemble. Car seulement un petit pourcentage du nouvel ensemble avec une annotation simple de deux classes peut donner jusqu'à 20% d'augmentation et ce genre d'annotation partielle peut se faire relativement rapidement en géomatique.

Un deuxième constat que l'on peut faire se trouve sur les écarts types des entraînements. La figure 4.7 nous avait montré des écarts types qui diminuaient en fonction du pourcentage de l'ensemble **P2** ajouté, mais ce n'est pas ce que nous retrouvons à la figure 4.9. Dans un premier cas, l'écart type ne semble pas diminuer avec le pour-

centage du jeu de données. De plus, la magnitude de l'écart type est beaucoup plus importante qu'à la figure 4.7, montrant une plus grande instabilité à travers les entraînements. On pouvait s'y attendre, pour 0% de **P1**, le réseau n'a vu que des images de Vaihingen, alors dépendamment de ce que le réseau a appris de ces images il généralisera différemment sur un nouvel ensemble de données. Bien que les écarts types soient plus petits lorsque l'on rajoute des images de Potsdam dans l'ensemble d'entraînement, ceux-ci restent quand même importants. Cela montre que l'ajout d'une annotation partielle augmente le *F1-score* moyen, mais ces prédictions sont instables selon l'entraînement contrairement à l'ajout d'une annotation complète.

Comme pour la première étape des expériences, on se demande si on obtient de meilleurs résultats sur les classes avec peu de pixels annotés en ajoutant plus de pixels d'un nouvel ensemble de données, mais cette fois avec une annotation partielle. Contrairement à la première expérience où nous regardions seulement une classe, on va observer comment chacune d'entre elles réagit à l'ajout d'un ensemble avec une annotation partielle (figure 4.10).

La première chose que la figure 4.10 nous montre est à quel point certaine classes sont moins affectées que d'autres par l'ajout de données partiellement annotées. On se rappelle que dans le tableau 4.3, on retrouve le pourcentage des pixels dans chaque classe. Dans la colonne de Vaihingen, on constate que 25.56% des pixels de l'ensemble d'entraînement appartiennent à la classe *routes*, il s'agit de la classe la mieux représentée dans Vaihingen et la deuxième dans Potsdam avec 24.58%. On ne s'attendait pas à ce que l'ajout de **P1** ait un si fort impact sur cette classe, pourtant dans la figure 4.10 la courbe de la classe *routes* augmente d'environ 40% après seulement 35% de l'ensemble **P1**. Cela veut dire que malgré ces connaissances sur l'ensemble de Vaihingen, le réseau à eu besoin d'aide pour bien généraliser les *routes* sur l'ensemble de Potsdam. Pourtant, si l'on prend la classe *bâtiments* qui fait partie du même ensemble hiérarchique que *routes* (figure 4.1), on n'observe pas le même comportement. La classe *bâtiments* qui représente 23.41% de Vaihingen et 22.91% de Potsdam, ne semble pas être très influencée par les données de **P1**. Bien que cette dernière augmente d'un peu plus de 10%, il lui faut au moins 71% des données **P1** pour atteindre

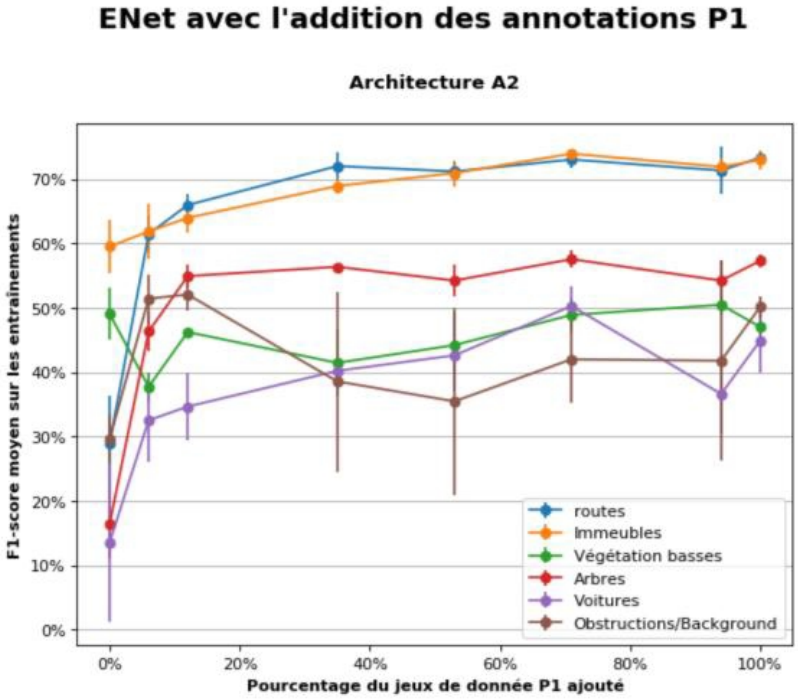


figure 4.10: Résultat moyen du *F1-score* de la moyenne des trois entraînements du modèle Enet[67] sur l'architectures $A2$ pour chauqe classes.

ce *F1-score*. Ce qui nous amène à penser que l'apprentissage des *bâtiments* se généralise bien d'un ensemble de données à un autre puisqu'ils sont généralement représentés sous la même forme. Continuons dans le groupe *fabrication humaine*, la classe *voitures* semble avoir un comportement semblable à la classe *bâtiments* où le simple fait d'ajouter 6% de **P1** augmente le *F1-score* de près de 15%. Par contre, il semble qu'à partir de 94% de **P1**, le réseau a plus de difficultés à généraliser les *voitures* passant d'un *F1-score* de 50% à moins de 40%. Il semble remonter la pente à 100% de **P1**, mais sans réatteindre le 50%. Nous n'avons malheureusement pas trouvé de raison qui pourrait expliquer ce phénomène. Il nous reste la courbe *background*, cette dernière est un peu spéciale, car elle ne suit aucun des comportements observés chez les autres classes. On peut toutefois remarquer qu'avec un peu de données **P1** (6%) on peut voir une amélioration. C'est lorsque l'on continue à en rajouter (dépassé 12% de **P1**), que le réseau semble confus, cela pourrait être dû au fait que les pixels de cette classe sont divers et qu'ajouter plus de pixels diminue le *F1-score*. Les écarts-types supportent

notre hypothèse, car ceux après 12% de **P1** sont très importants allant à prêt de 15%, cela semble se stabiliser lors de l'ajout complet de l'ensemble **P1**. Une autre classe qui semble avoir de la difficulté avec l'ajout de l'ensemble **P1** est *végétation basse*, le premier 6% de **P1** semble avoir confondu le réseau en perdant 10% du *F1-score*. Vers la fin de l'ajout de **P1** le *F1-score* semble toutefois être revenue à son point de départ avant l'ajout d'une image de Potsdam. Ce qui nous laisse avec la dernière classe, les *arbres*, il n'y a pas grand choses à dire de plus que la classe *voitures*, car les deux courbes ont des profiles quasi-identiques.

De façon générale, l'ajout d'annotations partielles (**P1**) aide à l'apprentissage profond et au transfert de connaissances pour un faible effort d'annotation. Cela dépend grandement des classes et de leurs relations avec l'annotation partielle que les ensembles de données partagés.

Comparaison des modèles Enet et Unet

Dans cette section nous comparons l'implémentation de notre architecture **A2** sur deux modèles déjà connus de la communauté, le Unet[73] (figure 2.5) et le Enet[67] (figure 2.9). Comme pour la partie "Transfert de connaissances avec annotation P1 seulement (Enet)", nous utilisons seulement **A2** pour des questions de rapidité et parce que l'architecture **A4** donne des résultats très similaires à **A2**. Nous mettons aussi **A1** de coté, car nous voulons comparer les performances de notre approche et non celles des modèles non modifiés.

Le déroulement de cette expérience ce déroule comme suit :

— On entraîne **A2** sur le Enet avec une annotation complète de Vaihingen pour ensuite prédire une annotation complète sur l'ensemble de test de Potsdam (0% de **P1** et **P2**). On répète l'expérience en ajoutant un certain pourcentage de l'ensemble d'entraînement Potsdam avec une annotation partielle (**P1**), utilisant la même gamme de pourcentages que lors de la première expérience.

— On entraîne **A2** sur le Unet avec une annotation complète de Vaihingen pour ensuite prédire une annotation complète sur l'ensemble de test de Potsdam (0% de **P1** et **P2**). Comme pour le Enet, on répète l'expérience en ajoutant un cer-

tain pourcentage de l'ensemble d'entraînement Potsdam avec une annotation partielle (**P1**).

— On entraîne **A2** sur le Enet avec une annotation complète et partielle de toutes les images de Vaihingen et toutes les images de Potsdam annotées partiellement **P1** (0% de **P2**). Ensuite, on prédit une annotation partielle et complète sur l'ensemble de test de Potsdam. Comme pour les étapes précédentes, on répète l'expérience avec un certain pourcentage de Potsdam, mais cette fois avec une annotation complète.

— On entraîne **A2** sur le Unet, avec une annotation complète et partielle de toutes les images de Vaihingen et toutes les images de Potsdam annotées partiellement **P1** (0% de **P2**). Ensuite, on prédit une annotation partielle et complète sur l'ensemble de test de Potsdam. Comme pour les étapes précédentes, on répète l'expérience avec un certain pourcentage de Potsdam, mais cette fois avec une annotation complète.

On répète l'ensemble de ces points trois fois pour obtenir une moyenne et un écart type du *F1-score* pour chaque test. Les résultats sont présentés dans les figures 4.11 et 4.12.

En premier lieu, on compare les deux réseaux sur l'ajout du jeu de données **P1**, comme pour la section précédente (figure 4.11). Le constat que l'on peut faire sur cette figure, est que le Enet[67] semble légèrement mieux performer que le Unet[73]. Dans son ensemble, le Unet[73] performe bien, celui-ci a un profil similaire avec le Enet[67], une forte augmentation dans les premiers ajouts de **P1** et se stabilise après 35% de **P1**. Par contre, le Enet[67] montre de résultats légèrement supérieurs, à l'exception du point à 94% de **P1** pour une raison que ne l'on peut pas expliquer. Si nous avions répété le processus 30 fois au lieu de seulement 3, les courbes auraient été plus lisses et ce type d'anormalités aurait sûrement disparue.

Contrairement à la figure 4.11, la figure 4.12 montre une légère supériorité pour le Unet[73], mais rien de significatif. De façon générale, on observe une forte augmentation dans les premiers 12% pour ensuite augmenter de façon moins agressive pour le reste de l'ajout de l'annotation complète de Potsdam **P2**.

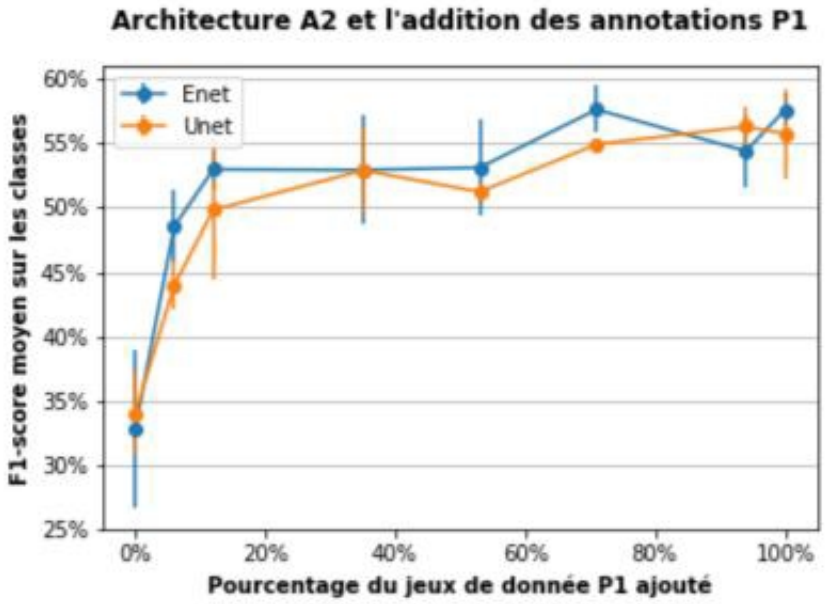


figure 4.11: Résultat moyen du *F1-score* de la moyenne des trois entraînements du modèle Enet[67] et Unet[73] sur l'architectures $A2$ avec les données **P1**.

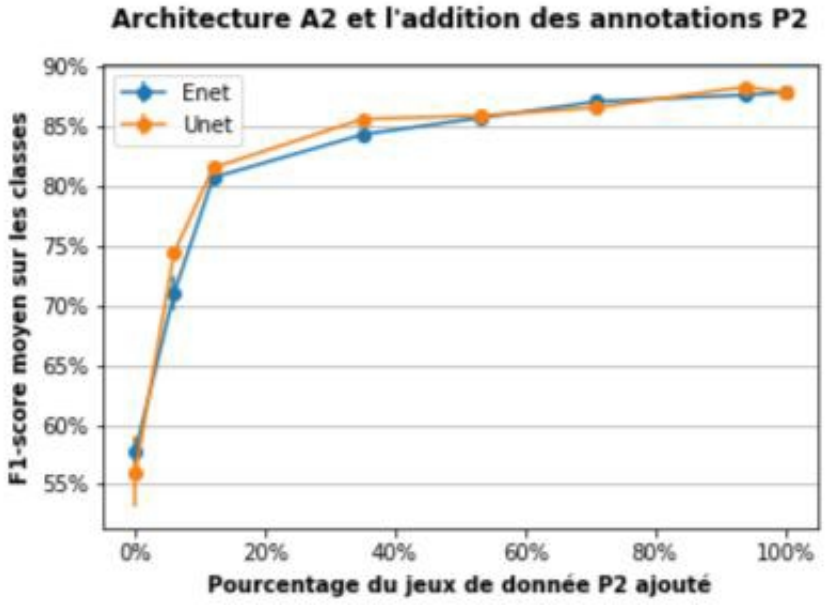


figure 4.12: Résultat moyen du *F1-score* de la moyenne des trois entraînements du modèle Enet[67] et Unet[73] sur l'architectures $A2$ avec les données **P2**.

Les courbes suivent une tendance logarithmique, autant pour l'ajout de $P1$ que de $P2$. Avec ces résultats, on ne peut pas statuer sur quel réseau performe le mieux, car les résultats des deux réseaux sont à toutes fins pratiques identiques. Toutefois, les résultats nous apprend que le simple fait d'apporter des modifications à une architecture comme **A2** à un modèle de segmentation permet l'incorporation de deux niveaux d'annotations et permet d'expecté des résultats se comportant comme les résultats obtenus dans ce chapitre. Même si nous n'en avons pas parlé dans l'analyse, le Enet[67] est catégoriquement plus rapide que le Unet[73] à l'entraînement et dans un cadre de recherche cela peut s'avouer un très grand avantage.

Conclusion

Cet ouvrage a présenté différentes façons d'approcher la segmentation d'images satellitaires via le concept de transfert de connaissances. L'analyse d'images satellites est avantageuse pour bien des raisons et les CNN permettent de tirer un maximum d'informations de ces vastes images. Les CNN sont des réseaux neuronaux capables d'apprendre des motifs sous forme de structures spatiales. Ce mémoire a présenté les fondements et les principes d'entraînement, pour ensuite en voir les différentes applications à travers de multiples expériences. Pour ce qui est du meilleur réseau dans ce genre de situation, notre choix serait le Enet[67]. Même s'il ne montre pas de résultats beaucoup plus élevés, il est catégoriquement plus rapide que le Unet[73] et dans le cadre de la recherche cela peut s'avouer un très grand avantage. Nous avons vu une panoplie de jeux de données dans lesquelles nous avons arrêté notre choix sur l'ensemble **ISPRS**, Vaihingen et Potsdam pour effectuer une étude préliminaire sur l'utilisation de l'apprentissage profond sur les images satellites.

À travers cette étude, on a abordé des questions telles que : *Quelle architecture de segmentation est la mieux adaptée pour segmenter des images de télédétection ?* ; *Les modalités d'entrée affectent-elles la précision de la segmentation ?* ; *Le choix d'une fonction de perte affecte-t-il la qualité globale des cartes de segmentation produites ?* ; *Combien d'images annotées sont nécessaires pour entraîner un modèle efficacement ?* ; *Est-ce que l'entraînement et la prédiction de zones d'images superposées ont une incidence sur les résultats obtenus ?* et *Qu'arrive-t-il lorsqu'un modèle entraîné sur un jeu de données "X" puis affiné sur un autre jeu de données "Y" ?*. Les résultats expérimentaux ont conduit aux conclusions suivantes :

1. Le réseau Unet semble la meilleure architecture, car elle est plus simple que

dmsmr, tout en étant légèrement plus précise.

2. Les CNN plus complexes (pyramidnet et tiramisu) ne semblent pas bien s'adaptés au contexte de l'imagerie satellitaire.
3. La fonction de perte "entropie croisée" est meilleure que la perte "Dice" ou une combinaison des deux.
4. L'utilisation de plusieurs modalités en entrée permet d'améliorer les résultats.
5. L'utilisation de correctifs de chevauchent d'image est une bonne solution pour augmenter le nombre données d'un ensemble de données satellitaires.
6. Malgré le fait que Potsdam soit un grand ensemble de données annotées, davantage de données d'entraînement pourrait conduire à de meilleurs résultats.
7. L'utilisation de poids préentraînés sur un ensemble de données et entraîner un modèle avec un autre ensemble de données est difficile en imagerie satellitaire. De plus, lorsqu'il s'agit d'un petit ensemble de données, la préformation d'un CNN sur un plus grand ensemble de données peut améliorer les résultats, en particulier sur les petites classes.

Ensuite, nous avons poursuivi nos expériences à l'utilisation d'images satellitaires partiellement annotées par des modèles d'apprentissage profond, amenant une nouvelle approche de type hiérarchique. Bien que les CNN soient hautement efficaces lorsque l'on utilise un seul ensemble de données et que l'on généralise sur ce même ensemble, ils ne sont généralement pas assez efficients pour généraliser sur un ensemble de données jamais vu et comportant de légères différences. Pour améliorer ce phénomène, on se tourne vers la combinaison d'ensembles de données. La combinaison permet d'optimiser un réseau neuronal, pour mieux généraliser sur des ensembles qui sont légèrement différents. Différentes approches de combinaison ont été explorées, dont l'approche d'annotation partielle qui permet une combinaison sur différents niveaux hiérarchiques. Ces méthodes ne performent pas mieux que la méthode originale $A1$, mais offrent un nouveau regard sur le nombre d'images annotées et la qualité de l'annotation requise pour bien performer. Une direction possible d'amélioration serait d'augmenter les niveaux hiérarchiques, passer de deux à trois ou quatre. Pour cela il faudrait trouver des ensembles de données avec un nombre de classes

plus élevé. Il reste de nombreuses questions méritant d'être étudiées dans des projets de recherche futurs. Le domaine du transfert de connaissance dans l'apprentissage profond requiert encore bon nombre d'études pour bien performer et généraliser. La taille d'un ensemble de données, la difficulté de la tâche, le temps d'entraînement et la structure du réseau sont parmi un nombre d'éléments interdépendants qui influencent l'efficience d'un réseau neuronal.

Des travaux permettant d'extraire de nouvelles classes sur d'anciens ensembles de données ou sur des images ailleurs dans le monde que le réseau n'a jamais vu seraient d'une grande utilité pour le domaine. Pour ce faire, le domaine des ensembles de données synthétiques semble une belle avenue à explorer, ce type de données permettraient de représenter des phénomènes qui sont très rares dans le domaine de l'imagerie satellitaire. Plusieurs articles ont déjà exploré cette idée en utilisant l'ensemble de données de GTA-V pour la conduite automatique[35, 49, 75], et dernièrement, un article est sorti utilisant le même jeu de données, mais cette fois vu du ciel[102]. Bien que cet article cherche seulement à faire du transfert de style, l'utilisation des images résultantes serait d'une grande utilité pour toute la communauté de l'imagerie satellitaire. Un ensemble de données où l'on peut contrôler tous les aspects de l'environnement et toutes les classes qui le composent.

Annexe A

Première annexe

Quelque définitions,

Réseau siamois Un réseau siamois est un réseau de neurones artificiels qui utilise les mêmes poids tout en travaillant en tandem sur deux vecteurs d'entrées différents pour calculer des vecteurs de sortie comparables. Souvent, l'un des vecteurs de sortie est précalculé, formant ainsi une ligne de base à laquelle l'autre vecteur de sortie est comparé.

Orthorectifié L'orthorectification est une correction géométrique des images qui a pour but de les présenter comme si elles avaient été acquises depuis la verticale. En pratique, il s'agit de rendre l'image acquise par le satellite superposable à une carte.

Résolution au sol La résolution au sol dans une image de la Terre prise dans les air, ou à partir de l'espace, est la distance entre les centres de pixels mesurée sur le terrain. Par exemple, dans une image d'un mètre de résolution au sol signifie que les pixels de l'image sont à une distance de un mètre sur le terrain.

Télédétection Science visant la mesure ou l'acquisition d'informations sur un objet ou un phénomène, par l'intermédiaire d'un instrument de mesure n'ayant pas de contact avec l'objet étudié.

Modèle numérique d'élévation Un Modèle Numérique d'Élévation (MNE) est une représentation des élévations sur un terrain comprenant les plantes et les bâtiments.

Annexe B

Deuxième annexe

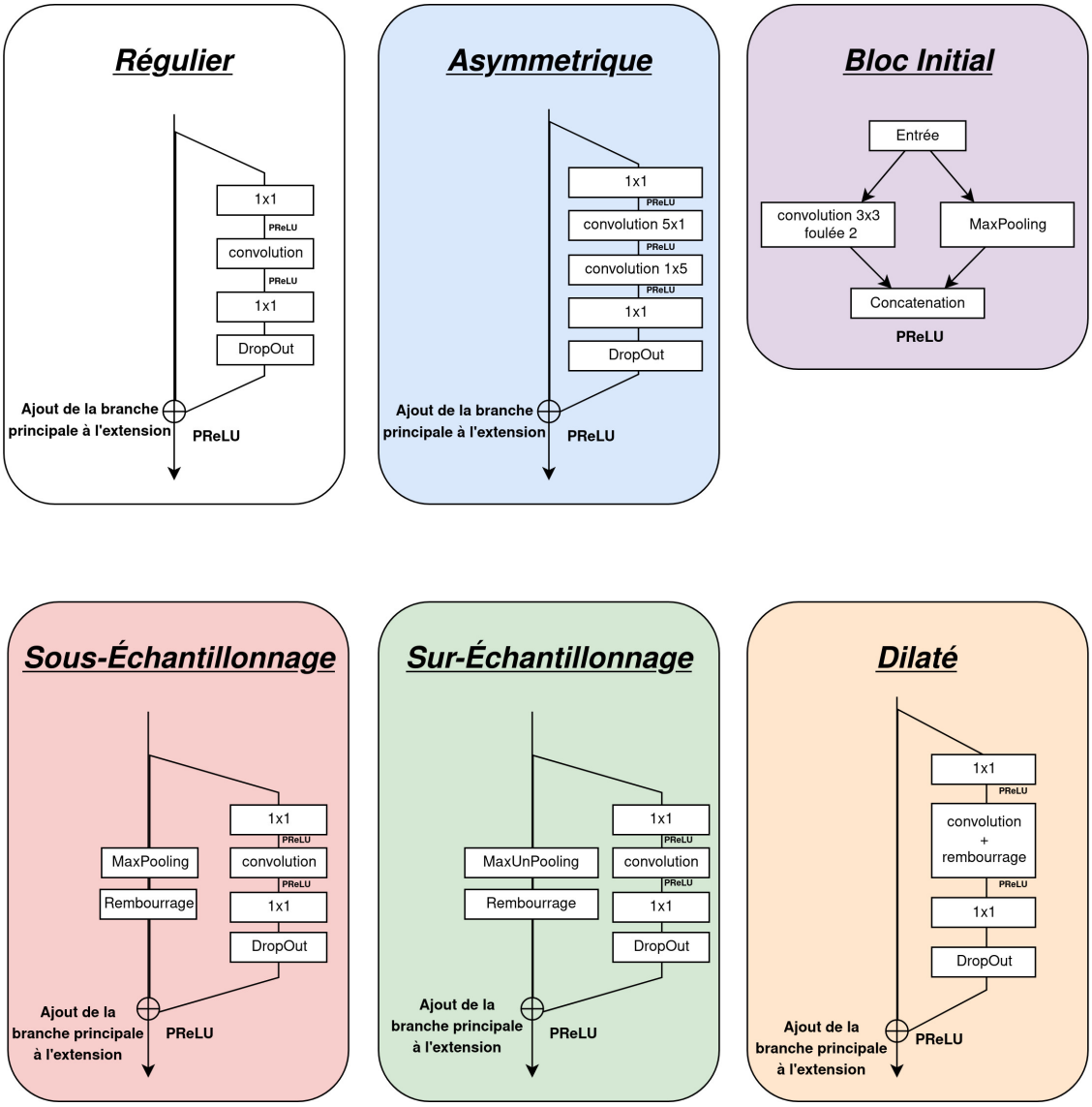


figure B.1: Détails des différentes couches utilisées.

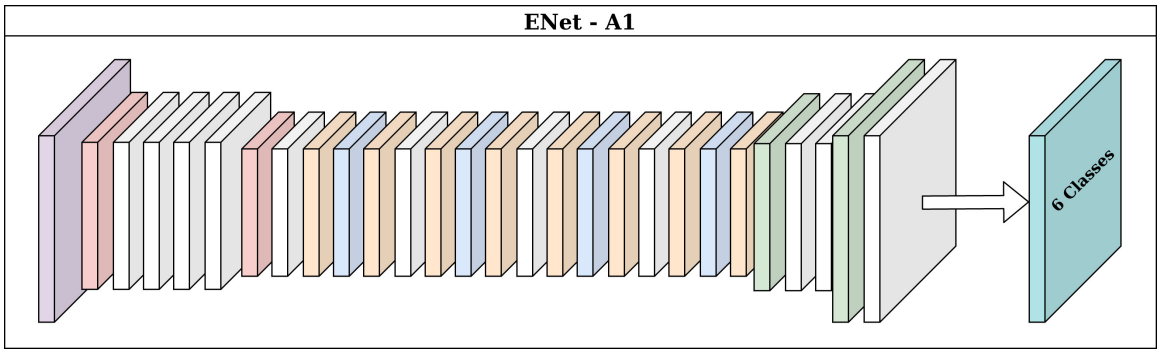


figure B.2: Modèle Enet[67] avec l'architecture **A1**.

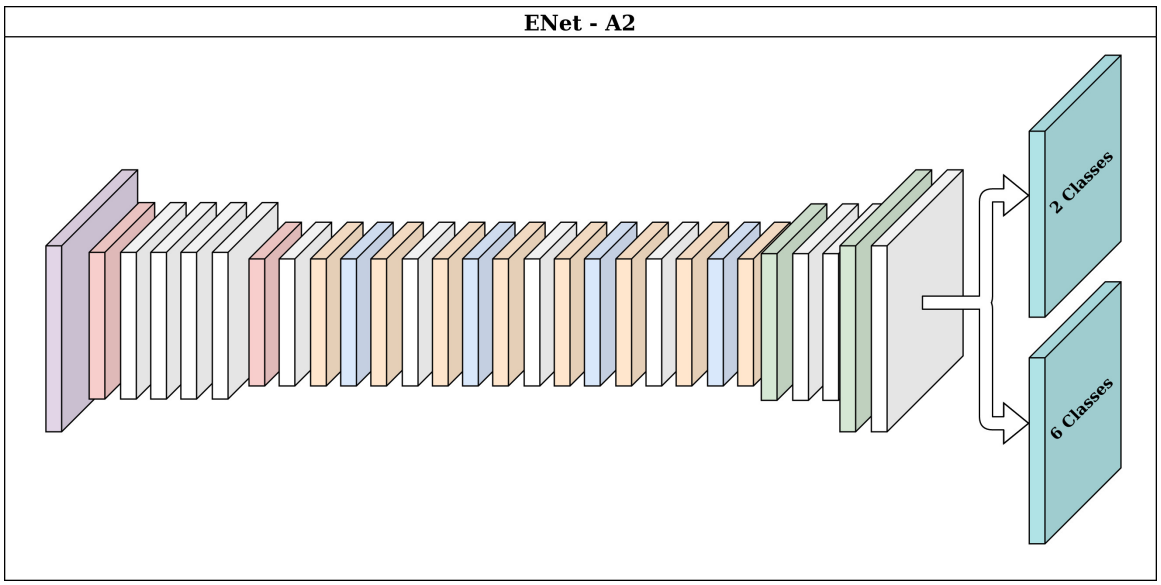


figure B.3: Modèle Enet[67] avec modification de l'architecture **A2**.

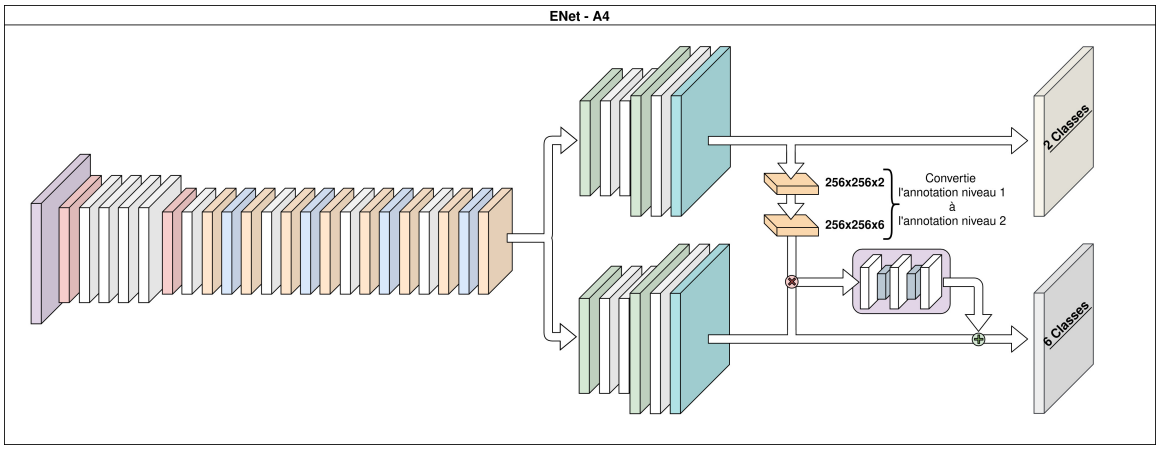


figure B.4: Modèle Enet[67] avec modification de l'architecture **A4**.

Bibliographie

[1] Seyed Ali Ahmadi et Ali Mohammadzadeh.
« A simple method for detecting and tracking vehicles and vessels from high resolution spaceborne videos ».
Dans *2017 Joint Urban Remote Sensing Event (JURSE)*, pages 1–4. IEEE, 2017.

[2] Amir Akbarzadeh, J-M Frahm, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, Paul Merrell, M Phelps, S Sinha, B Talton et others.
« Towards urban 3d reconstruction from video ».
Dans *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 1–8. IEEE, 2006.

[3] Jiří Apeltauer, Adam Babinec, David Herman et Tomáš Apeltauer.
« Automatic vehicle trajectory extraction for traffic analysis from aerial video data ».
The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 40(3):9, 2015.

[4] Nicolas Audebert, Bertrand Le Saux et Sébastien Lefèvre.
« Beyond RGB : Very high resolution urban remote sensing with multimodal deep networks ».
ISPRS Journal of Photogrammetry and Remote Sensing, 140:20–32, 2018.

[5] Carlos Lima Azevedo, João L Cardoso, Moshe Ben-Akiva, João P Costeira et Manuel Marques.
« Automatic vehicle trajectory extraction by aerial remote sensing ».
Procedia-Social and Behavioral Sciences, 111:849–858, 2014.

[6] Vijay Badrinarayanan, Alex Kendall et Roberto Cipolla.
« SegNet : A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation ».
IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12):2481–2495, Dec 2017.

[7] John E Ball, Derek T Anderson et Chee Seng Chan.
« Comprehensive survey of deep learning in remote sensing : theories, tools, and challenges for the community ».
Journal of Applied Remote Sensing, 11(4), 2017.

[8] Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki et Ramakrishna Nemani.
« DeepSat ».
Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2015.

[9] Yakoub Bazi, Farid Melgani et Hamed D Al-Sharari.
« Unsupervised change detection in multispectral remotely sensed imagery with level set methods ».
IEEE Transactions on Geoscience and Remote Sensing, 48(8):3178–3187, 2010.

[10] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger et Roopak Shah.
« Signature verification using a" siamese" time delay neural network ».
Dans *Advances in neural information processing systems*, pages 737–744, 1994.

[11] Lorenzo Bruzzone et Diego F Prieto.
« Automatic analysis of the difference image for unsupervised change detection ».
IEEE Transactions on Geoscience and Remote sensing, 38(3):1171–1182, 2000.

[12] G. Camps-Valls, D. Tuia, L. Bruzzone et J. A. Benediktsson.
« Advances in Hyperspectral Image Classification : Earth Monitoring with Statistical Learning Methods ».
IEEE Signal Processing Magazine, 31(1):45–54, Jan 2014.

[13] Guo Cao, Licun Zhou et Yupeng Li.
« A new change-detection method in high-resolution remote sensing images

based on a conditional random field model ».
International Journal of Remote Sensing, 37(5):1173–1189, 2016.

[14] Turgay Celik.
« Unsupervised change detection in satellite images using principal component analysis and k-means clustering ».
IEEE Geoscience and Remote Sensing Letters, 6(4):772–776, 2009.

[15] G. Cheng, J. Han et X. Lu.
« Remote Sensing Image Scene Classification : Benchmark and State of the Art ».
Proceedings of the IEEE, 105(10):1865–1883, Oct 2017.

[16] Gong Cheng, Junwei Han et Xiaoqiang Lu.
« Remote Sensing Image Scene Classification : Benchmark and State of the Art ».
Proceedings of the IEEE, 105(10):1865–1883, Oct 2017.

[17] Gordon Christie, Neil Fendley, James Wilson et Ryan Mukherjee.
« Functional Map of the World ».
2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6172–6180, Jun 2018.

[18] Diana Contreras, Thomas Blaschke, Dirk Tiede et Marianne Jilge.
« Monitoring recovery after earthquakes through the integration of remote sensing, GIS, and ground observations : the case of L'Aquila (Italy) ».
Cartography and Geographic Information Science, 43(2):115–133, 2016.

[19] Pol Coppin, Inge Jonckheere, Kristiaan Nackaerts, Bart Muys et Eric Lambin.
« Review ArticleDigital change detection methods in ecosystem monitoring : a review ».
International journal of remote sensing, 25(9):1565–1596, 2004.

[20] William R Crum, Oscar Camara et Derek LG Hill.
« Generalized overlap measures for evaluation and validation in medical image analysis ».
IEEE transactions on medical imaging, 25(11):1451–1461, 2006.

[21] Zhipeng Deng, Hao Sun, Shilin Zhou, Juanping Zhao, Lin Lei et Huanxin Zou.
« Multi-scale object detection in remote sensing imagery with convolutional neural networks ».
ISPRS journal of photogrammetry and remote sensing, 145:3–22, 2018.

[22] Foivos I Diakogiannis, François Waldner, Peter Caccetta et Chen Wu.
« Resunet-a : a deep learning framework for semantic segmentation of remotely sensed data ».
ISPRS Journal of Photogrammetry and Remote Sensing, 162:94–114, 2020.

[23] Peng Ding, Ye Zhang, Wei-Jian Deng, Ping Jia et Arjan Kuijper.
« A light and faster regional convolutional neural network for object detection in optical remote sensing images ».
ISPRS journal of photogrammetry and remote sensing, 141:208–218, 2018.

[24] Pedram Ghamisi, Yushi Chen et Xiao Zhu.
« A Self-Improving Convolution Neural Network for the Classification of Hyperspectral Data ».
IEEE Geoscience and Remote Sensing Letters, 13:1–5, 10 2016.

[25] Nicolas Girard, Guillaume Charpiat et Yuliya Tarabalka.
« Aligning and updating cadaster maps with aerial images by multi-task, multi-resolution deep learning ».
Dans *Asian Conference on Computer Vision*, pages 675–690. Springer, 2018.

[26] Ritwik Gupta, Richard Hosfelt, Sandra Sajeev, Nirav Patel, Bryce Goodman, Jigar Doshi, Eric Heim, Howie Choset et Matthew Gaston.
« xbd : A dataset for assessing building damage from satellite imagery ».
2019.

[27] Martin Habbecke et Leif Kobbelt.
« Automatic registration of oblique aerial images with cadastral maps ».
Dans *European Conference on Computer Vision*, pages 253–266. Springer, 2010.

[28] Geoffrey J Hay, Guillermo Castilla, Michael A Wulder et Jose R Ruiz.
« An automated object-based approach for the multiscale image segmentation of forest scenes ».

International Journal of Applied Earth Observation and Geoinformation, 7(4):339–359, 2005.

[29] Haiqing He, Min Chen, Ting Chen et Dajun Li.
« Matching of remote sensing images with complex background variations via Siamese convolutional neural network ».
Remote Sensing, 10(2):355–355, 2018.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun.
« Deep residual learning for image recognition ».
Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun.
« Deep residual learning for image recognition ».
Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Patrick Helber, Benjamin Bischke, Andreas Dengel et Damian Borth.
« EuroSAT : A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification ».
IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 12(7):2217–2226, Jul 2019.

[33] Bohao Huang, Daniel Reichman, Leslie M Collins, Kyle Bradbury et Jordan M Malof.
« Tiling and Stitching Segmentation Output for Remote Sensing : Basic Challenges and Recommendations ».
arXiv preprint arXiv :1805.12219, 2018.

[34] Gao Huang, Zhuang Liu, Laurens Van Der Maaten et Kilian Q Weinberger.
« Densely connected convolutional networks ».
pages 4700–4708, 2017.

[35] Sheng-Wei Huang, Che-Tsung Lin, Shu-Ping Chen, Yen-Yi Wu, Po-Hao Hsu et Shang-Hong Lai.
« Auggan : Cross domain adaptation with gan-based data augmentation ».

Dans *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 718–731, 2018.

[36] Lloyd Hughes, Michael Schmitt et Xiao Zhu.
« Mining hard negative samples for sar-optical image matching using generative adversarial networks ».
Remote Sensing, 10(10):1552–1552, 2018.

[37] Lloyd H Hughes, Michael Schmitt, Lichao Mou, Yuanyuan Wang et Xiao Xiang Zhu.
« Identifying corresponding patches in SAR and optical images with a pseudo-siamese CNN ».
IEEE Geoscience and Remote Sensing Letters, 15(5):784–788, 2018.

[38] Vladimir Iglovikov et Alexey Shvets.
« TernausNet : U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation ».
arXiv preprint arXiv :1801.05746, 2018.

[39] Mahdi Javanmardi, Yanlei Gu, Ehsan Javanmardi, Li-Ta Hsu et Shunsuke Kamijo.
« 3D building map reconstruction in dense urban areas by integrating airborne laser point cloud with 2D boundary map ».
Dans *2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 126–131. IEEE, 2015.

[40] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero et Yoshua Bengio.
« The one hundred layers tiramisu : Fully convolutional densenets for semantic segmentation ».
Dans *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1175–1183. IEEE, 2017.

[41] Ping Jian, Keming Chen et Chenwei Zhang.
« A hypergraph-based context-sensitive representation technique for VHR remote-sensing image change detection ».
International Journal of Remote Sensing, 37(8):1814–1825, 2016.

[42] Nikolai Vladimirovich Kim et Mikhail Alekseevich Chervonenkis.
« Situation control of unmanned aerial vehicles for road traffic monitoring ».
Modern Applied Science, 9(5):1–1, 2015.

[43] Martin Kirscht et Carsten Rinke.
« 3D Reconstruction of Buildings and Vegetation from Synthetic Aperture Radar (SAR) Images ».
Dans *In Proceedings of IAPR Workshop on Machine Vision Applications*, pages 17–19, 1998.

[44] George Kopsiaftis et Konstantinos Karantzalos.
« Vehicle detection and traffic density monitoring from very high resolution satellite video data ».
Dans *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1881–1884. IEEE, 2015.

[45] Alex Krizhevsky.
« One weird trick for parallelizing convolutional neural networks ».
arXiv preprint arXiv :1404.5997, 2014.

[46] John Lafferty, Andrew McCallum et Fernando CN Pereira.
« Conditional random fields : Probabilistic models for segmenting and labeling sequence data ».
2001.

[47] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov et Brendan McCord.
« xview : Objects in context in overhead imagery ».
arXiv preprint arXiv :1802.07856, 2018.

[48] Charis Lanaras, José Bioucas-Dias, Silvano Galliani, Emmanuel Baltsavias et Konrad Schindler.
« Super-resolution of Sentinel-2 images : Learning a globally applicable deep neural network ».
ISPRS Journal of Photogrammetry and Remote Sensing, 146:305–319, 2018.

[49] Peilun Li, Xiaodan Liang, Daoyuan Jia et Eric P Xing.
« Semantic-aware grad-gan for virtual-to-real urban scene adaption ».
arXiv preprint arXiv :1801.01726, 2018.

[50] Xianju Li, Xinwen Cheng, Weitao Chen, Gang Chen et Shengwei Liu.
« Identification of forested landslides using LiDar data, object-based image analysis, and machine learning algorithms ».
Remote sensing, 7(8):9705–9726, 2015.

[51] Xiaoxiao Li, Soe W Myint, Yujia Zhang, Chritopher Galletti, Xiaoxiang Zhang et Billie L Turner II.
« Object-based land-cover classification for metropolitan Phoenix, Arizona, using aerial photography ».
International Journal of Applied Earth Observation and Geoinformation, 33:321–330, 2014.

[52] Yansheng Li, Xin Huang et Hui Liu.
« Unsupervised deep feature learning for urban village detection from high-resolution remote sensing images ».
Photogrammetric Engineering & Remote Sensing, 83(8):567–579, 2017.

[53] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie.
« Feature pyramid networks for object detection ».
pages 2117–2125, 2017.

[54] Y. Liu, S. Piramanayagam, S. T. Monteiro et E. Saber.
« Dense Semantic Labeling of Very-High-Resolution Aerial Imagery and LiDAR with Fully-Convolutional Neural Networks and Higher-Order CRFs ».
Dans *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1561–1570, 2017.

[55] Yongcheng Liu, Bin Fan, Lingfeng Wang, Jun Bai, Shiming Xiang et Chunhong Pan.
« Semantic labeling in very high resolution images via a self-cascaded convolutional neural network ».
ISPRS Journal of Photogrammetry and Remote Sensing, 145:78 – 95, 2018.
Deep Learning RS Data.

[56] Yu Liu, Duc Minh Nguyen, Nikos Deligiannis, Wenrui Ding et Adrian Munteanu.
« Hourglass-shapenetwork based semantic segmentation for high resolution aerial imagery ».
Remote Sensing, 9(6):522–522, 2017.

[57] Efren López-Jiménez, Juan Irving Vasquez-Gomez, Miguel Angel Sanchez-Acevedo, Juan Carlos Herrera-Lozada et Abril Valeria Uriarte-Arcia.
« Columnar cactus recognition in aerial images using a deep learning approach ».
Ecological Informatics, 52:131 – 138, 2019.

[58] Dimitrios Marmanis, Mihai Datcu, Thomas Esch et Uwe Stilla.
« Deep learning earth observation classification using ImageNet pretrained networks ».
IEEE Geoscience and Remote Sensing Letters, 13(1):105–109, 2015.

[59] Nina Merkle, Stefan Auer, Rupert Müller et Peter Reinartz.
« Exploring the potential of conditional adversarial networks for optical and SAR image matching ».
IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 11(6):1811–1820, 2018.

[60] Nina Merkle, Wenjie Luo, Stefan Auer, Rupert Müller et Raquel Urtasun.
« Exploiting deep matching and SAR data for the geo-localization accuracy improvement of optical satellite images ».
Remote Sensing, 9(6):586–586, 2017.

[61] S. Mohajerani, T. A. Krammer et P. Saeedi.
« "A Cloud Detection Algorithm for Remote Sensing Images Using Fully Convolutional Neural Networks" ».
Dans *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5, Aug 2018.

[62] Lichao Mou et Xiao Xiang Zhu.
« Spatiotemporal scene interpretation of space videos via deep neural network and tracklet analysis ».

Dans *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1823–1826. IEEE, 2016.

[63] T Nathan Mundhenk, Goran Konjevod, Wesam A Sakla et Kofi Boakye.
« A large contextual dataset for classification, detection and counting of cars with deep learning ».
Dans *European Conference on Computer Vision*, pages 785–800. Springer, 2016.

[64] Hyeonwoo Noh, Seunghoon Hong et Bohyung Han.
« Learning deconvolution network for semantic segmentation ».
Dans *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[65] Ender Oguslu.
« Sparse coding based feature representation method for remote sensing images ».

[66] Xuran Pan, Lianru Gao, Bing Zhang, Fan Yang et Wenzhi Liao.
« High-Resolution Aerial Imagery Semantic Labeling with Dense Pyramid Network ».
Sensors, 18(11):3774–3774, Nov 2018.

[67] Adam Paszke, Abhishek Chaurasia, Sangpil Kim et Eugenio Culurciello.
« Enet : A deep neural network architecture for real-time semantic segmentation ».
arXiv preprint arXiv :1606.02147, 2016.

[68] Artzai Picon, Ovidiu Ghita, Paul F. Whelan et Pedro M. Iriondo.
« Spectral and Spatial Feature Integration for Classification of Nonferrous Materials in Hyperspectral Data ».
IEEE Transactions on Industrial Informatics, 5(4):483–494, novembre 2009.

[69] Michael Pidwirny.
Dans *Fundamentals of Physical Geography*, chapitre 2 et 8.
PhysicalGeography.net, 2007.

[70] Sankaranarayanan Piramanayagam, Eli Saber, Wade Schwartzkopf et Frederick Koehler.
« Supervised Classification of Multisensor Remotely Sensed Images Using a

Deep Learning Framework ».
Remote Sensing, 10(9):1429–1429, Sep 2018.

[71] Lingyan Ran, Yanning Zhang, Wei Wei et Qilin Zhang.
« A Hyperspectral Image Classification Framework with Spatial Pixel Pair Features ».
Sensors, 17(10):2421–2421, octobre 2017.

[72] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi et Silvio Savarese.
« Learning social etiquette : Human trajectory understanding in crowded scenes ».
Dans *European conference on computer vision*, pages 549–565. Springer, 2016.

[73] Olaf Ronneberger, Philipp Fischer et Thomas Brox.
« U-net : Convolutional networks for biomedical image segmentation ».
pages 234–241, 2015.

[74] Frank Rosenblatt.
« The perceptron : a probabilistic model for information storage and organization in the brain. ».
Psychological review, 65(6):386–386, 1958.

[75] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim et Rama Chellappa.
« Learning from synthetic data : Addressing domain shift for semantic segmentation ».
Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3752–3761, 2018.

[76] Teresa Santos, José Tenedório, Jorge Rocha et S. ENCARNAÇÃO.
« SATSTAT : Exploratory Analysis of Envisat-MERIS Data for Land Cover Mapping of Portugal in 2003 ».
2005.

[77] Zhenfeng Shao, Ke Yang et Weixun Zhou.
« Performance evaluation of single-label and multi-label remote sensing image retrieval using a dense labeling dataset ».
Remote Sensing, 10(6):964–964, 2018.

[78] Jamie Sherrah.
« Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery ».
arXiv preprint arXiv :1606.02585, 2016.

[79] Karen Simonyan et Andrew Zisserman.
« Very deep convolutional networks for large-scale image recognition ».
arXiv preprint arXiv :1409.1556, 2014.

[80] GEORGE W. STIMSON.
« Introduction to AIRBORNE RADAR ».
chapitre 1 - Basic Concepts.
SciTech Publishing, Inc., 1998.

[81] Gencer Sumbul, Marcela Charfuelan, Begüm Demir et Volker Markl.
« Bigearthnet : A large-scale benchmark archive for remote sensing image understanding », 2019.

[82] Xiaofeng Sun, Shuhan Shen et Zhanyi Hu.
« Automatic building extraction from oblique aerial images ».
Dans *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 663–668. IEEE, 2016.

[83] Vladimir Vapnik.
« Principles of risk minimization for learning theory ».
Dans *Advances in neural information processing systems*, pages 831–838, 1992.

[84] Michele Volpi et Devis Tuia.
« Dense semantic labeling of subdecimeter resolution images with convolutional neural networks ».
IEEE Transactions on Geoscience and Remote Sensing, 55(2):881–893, 2016.

[85] Hongzhen Wang, Ying Wang, Qian Zhang, Shiming Xiang et Chunhong Pan.
« Gated convolutional neural network for semantic segmentation in high-resolution images ».
Remote Sensing, 9(5):446–446, 2017.

[86] Shaona Wang, Shuyuan Yang et Licheng Jiao.
« Saliency-guided change detection for SAR imagery using a semi-supervised Laplacian SVM ».
Remote sensing letters, 7(11):1043–1052, 2016.

[87] Shuang Wang, Dou Quan, Xuefeng Liang, Mengdan Ning, Yanhe Guo et Licheng Jiao.
« A deep learning framework for remote sensing image registration ».
ISPRS Journal of Photogrammetry and Remote Sensing, 145:148–164, 2018.

[88] Chen Wu, Bo Du, Xiaohui Cui et Liangpei Zhang.
« A post-classification change detection method based on iterative slow feature analysis and Bayesian soft fusion ».
Remote Sensing of Environment, 199:241–255, 2017.

[89] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo et Liangpei Zhang.
« DOTA : A Large-Scale Dataset for Object Detection in Aerial Images ».
2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3974–3983, 6 2018.

[90] Tao Yang, Xiwen Wang, Bowei Yao, Jing Li, Yanning Zhang, Zhannan He et Wencheng Duan.
« Small moving vehicle detection in a satellite video of an urban area ».
Sensors, 16(9):1528–1528, 2016.

[91] Yi Yang et Shawn Newsam.
« Bag-of-visual-words and spatial extensions for land-use classification ».
Dans *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*, pages 270–279. ACM, 2010.

[92] Yun Yang, Alfred Stein, Valentyn A Tolpekin et Yang Zhang.
« High-Resolution Remote Sensing Image Classification Using Associative Hierarchical CRF Considering Segmentation Quality ».
IEEE Geoscience and Remote Sensing Letters, 15(5):754–758, 2018.

[93] Yuanxin Ye, Jie Shan, Lorenzo Bruzzone et Li Shen.
« Robust registration of multimodal remote sensing images based on structural

similarity ».
IEEE Transactions on Geoscience and Remote Sensing, 55(5):2941–2958, 2017.

[94] Armand Zampieri, Guillaume Charpiat, Nicolas Girard et Yuliya Tarabalka.
« Multimodal image alignment through a multiscale chain of neural networks with application to remote sensing ».
Dans *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 657–673, 2018.

[95] Mi Zhang, Xiangyun Hu, Like Zhao, Ye Lv, Min Luo et Shiyan Pang.
« Learning dual multi-scale manifold ranking for semantic segmentation of high-resolution images ».
Remote Sensing, 9(5):500–500, 2017.

[96] Ji Zhao, Yanfei Zhong et Liangpei Zhang.
« Detail-preserving smoothing classifier based on conditional random fields for high spatial resolution remote sensing imagery ».
IEEE transactions on geoscience and remote sensing, 53(5):2440–2452, 2015.

[97] Ping Zhong et Runsheng Wang.
« A multiple conditional random fields ensemble model for urban area detection in remote sensing optical images ».
IEEE Transactions on Geoscience and Remote Sensing, 45(12):3978–3988, 2007.

[98] Yanfei Zhong, Xiaobing Han et Liangpei Zhang.
« Multi-class geospatial object detection based on a position-sensitive balancing framework for high spatial resolution remote sensing imagery ».
ISPRS journal of photogrammetry and remote sensing, 138:281–294, 2018.

[99] Xiaoxiang Zhu, Jingliang Hu, Chunping Qiu, Yilei Shi, Hossein Bagheri, Jian Kang, Hao Li, Lichao Mou, Guicheng Zhang, Matthias Häberle, Shiyao Han, Yuansheng Hua, Rong Huang, Lloyd Hughes, Yao Sun, Michael Schmitt et Yuanyuan Wang.
« So2Sat LCZ42 », 2018.

[100] Barbara Zitova et Jan Flusser.
« Image registration methods : a survey ».
Image and vision computing, 21(11):977–1000, 2003.

[101] Qin Zou, Lihao Ni, Tong Zhang et Qian Wang.
« Deep learning based feature selection for remote sensing scene classification ».
IEEE Geoscience and Remote Sensing Letters, 12(11):2321–2325, 2015.

[102] Zhengxia Zou, Tianyang Shi, Wenyuan Li, Zhou Zhang et Zhenwei Shi.
« Do Game Data Generalize Well for Remote Sensing Image Segmentation ? ».
Remote Sensing, 12(2):275–275, Jan 2020.